Control de Inventario con Remanufacturación y Clasificación de Retornos

Informe Final

de

Proyecto de grado

Luciano Alvarez
Gabriela Arriola
Matilde Macciò

Supervisor: Pedro Piñeyro

Resumen

Este proyecto trata sobre la optimización de un problema de control de inventario con remanufacturación, clasificación, venta y disposición de retornos. En la literatura consultada al inicio del proyecto no se encontró un problema que reuniera estas características.

El alcance abarca la construcción de un modelo matemático para el problema mencionado, el desarrollo de una metaheurística para resolverlo y la comparación de sus resultados contra los de una herramienta de resolución exacta donde se modela el mismo problema.

Como metaheurística se desarrolló un algoritmo genético, implementado en la biblioteca ECJ de computación evolutiva escrita en Java. Como herramienta de resolución exacta se utilizó GLPK.

Se generaron juegos de datos basados en investigaciones ya existentes, y adaptados a las características de este problema. Se compararon y analizaron los resultados de los mismos luego de ejecutarlos con las herramientas mencionadas anteriormente.

Con el algoritmo genético se obtuvieron soluciones cercanas a las obtenidas con GLPK dentro del rango aceptable en la mayoría de los casos.

Palabras clave: inventario, producción, remanufacturación, clasificación, disposición final, algoritmo genético.

Tabla de Contenido

1. Introducción

El objetivo de éste documento es describir el trabajo realizado en el marco del proyecto de grado, sobre: la construcción de un modelo matemático para un problema de inventario con remanufacturación, clasificación, venta y disposición final de retornos, el desarrollo de un algoritmo genético para resolverlo y la comparación de sus resultados contra los de una herramienta de optimización que modela el mismo problema.

1.1. Descripción del problema

En el área industrial y comercial, algunos de los factores que generan la necesidad de controlar los niveles de inventario son: variaciones en la demanda, limitaciones en la capacidad de producción, limitaciones de espacio en depósitos, tiempos y costos de poner en producción un tipo de artículo. El control de inventario ayuda a determinar cuándo y con cuánto abastecer el inventario de un artículo para satisfacer su demanda, minimizando la suma de los costos involucrados, maximizando así las ganancias [11].

Debido a la complejidad y variabilidad que introducen factores como los antes mencionados, es importante contar con una herramienta que ayude a determinar estas interrogantes y que además lo haga en un tiempo razonable.

El Problema de Tamaño de Lote Económico (ELSP) hace referencia al problema de determinar cuándo y cuánto producir u ordenar de un artículo para satisfacer los requerimientos de la demanda, minimizando la suma de los costos involucrados. Generalmente se asume que los valores de la demanda son conocidos y que pueden variar en los diferentes períodos dentro de un horizonte de planificación finito.

El Problema de Tamaño de Lote Económico con Remanufacturación (ELSR) extiende el problema ELSP considerando remanufacturación de retornos. La remanufacturación es un proceso de recuperación de artículos usados y retornados (denominados comúnmente como retornos) para devolverlos al menos a su estado original y que de esta manera ofrezcan las mismas prestaciones que un artículo nuevo. Esto puede involucrar tareas como desmontaje, limpieza, remplazo de partes, reparación, montaje y testeo [23]. Los beneficios son varios: para el fabricante, permite reducir costos de operación, consumo de energía y niveles de contaminación, ya que se reutiliza el retorno en lugar de desecharlo, ayudando a cumplir con parte de la demanda. El cliente podría llegar a verlo reflejado en una reducción del precio del producto.

En los últimos años muchas empresas se han visto involucradas en la remanufacturación y disposición final de retornos, debido a que no son ajenas a

cuidar el medioambiente, ya sea por compromiso social o porque existen entidades reguladoras que los obligan a tener un cuidado especial hacia el entorno. Además, se debe tener en cuenta el hecho de que cuando una empresa decide no remanufacturar puede hacer que los retornos se dirijan a otras empresas, donde sí sean remanufacturados, afectando la cuota de mercado y los ingresos del fabricante original [13].

Generalmente la cantidad de productos retornados a remanufacturar no alcanza a cubrir la demanda, por lo cual es importante poder llevar adelante y coordinar ambas tareas: producción y remanufacturación. La planificación de las mismas puede ser una tarea compleja si deben compartir los recursos (por ejemplo, personal o infraestructura) [19].

Los retornos deben reunir ciertas características para que puedan ser remanufacturados [9]. Es necesario que pasen por un proceso de inspección y posterior clasificación ya que el nivel de calidad con el que llegan puede ser muy variado y afectar por lo tanto las tareas, y por ende el costo de la remanufacturación. En caso de que los retornos no cumplan con los requisitos mínimos exigidos para que puedan ser remanufacturados, los mismos pueden ser vendidos en el mercado o desechados de forma adecuada (disposición final) con el objetivo de minimizar el impacto ambiental.

En este proyecto de grado se considera el ELSR con clasificación de retornos (ELSRC), el cual agrega la característica de contar con retornos previamente clasificados según su nivel de calidad. Además, para aquellos retornos que no puedan ser remanufacturados se incluye la opción de que puedan ser vendidos en el mercado o desechados de forma adecuada (disposición final). El nivel de calidad de los retornos impacta directamente sobre los costos de remanufacturación y venta, dado que un retorno de peor calidad requiere un mayor trabajo de recuperación.

Entre la literatura consultada durante la investigación, no se encontró un problema con estas características.

Una de las empresas con más relevancia en el mundo de la remanufacturación es Caterpillar [8], líder global en el negocio de la remanufactura, con locaciones en más de diez países. Su programa CatReman remanufactura más de 2 millones de retornos por año (68 millones de kilogramos de material reciclado), renovando la vida útil de los productos. Esto favorece el medio ambiente, ya que en el proceso de remanufacturación se reducen desperdicios, producción de gases de efecto invernadero y la necesidad de materia prima.

Otra de las empresas involucradas en la remanufacturación desde principios de 1990 es Xerox. Ha sido pionera en la remanufacturación de

productos al final de su vida útil, convirtiéndolos en productos nuevos con la misma calidad y rendimiento que los manufacturados, respetando sus iniciativas de reciclaje sin desperdicios. Desde la fase de diseño las máquinas están diseñadas con el mínimo número de piezas, de alta durabilidad y capacidad de reutilización, motivando a múltiples ciclos de vida para el producto. En 2010 Xerox logró evitar 46 mil toneladas de desperdicio gracias a sus programas de remanufacturación y reciclaje [26].

La empresa alemana BMW también ha incorporado el proceso de remanufacturación y disposición final a la fabricación de automóviles. En 2006, logró reducir sus desperdicios un 45% debido a la utilización de material reciclado para la elaboración de los autos. Y en 2015, alcanzó a desechar solo 4kg de material por automóvil producido [7].

A nivel nacional, existe una iniciativa de recuperación de artículos propuesta por la empresa Unilever [24]. Unilever es una compañía internacional dedicada a la fabricación de insumos de limpieza para el hogar, cuidado personal y alimentos. Se planteó como desafío el envío de cero residuos a relleno sanitario, que pueden generar colapso de materiales enterrados, proliferación de plagas y lixiviados hacia zonas residenciales. Estos residuos provienen por ejemplo de disposición de alimentos vencidos, devoluciones de productos averiados, y de las propias plantas de producción. La alternativa encontrada hasta el momento para disponer estos residuos minimizando el impacto ambiental, es el compostaje de los mismos, a pesar de ser un proceso costoso [25].

1.2.Organización del documento

El resto del documento se estructura de la siguiente manera. En la Sección 2 se hace referencia al relevamiento de la literatura realizado sobre el tema, además de presentar algunos términos importantes para la comprensión del problema y el resto del documento. En la Sección 3 se presentan las características del problema estudiado y el modelo matemático. En la Sección 4 se explican las implementaciones realizadas: implementación del modelo con GLPK, del algoritmo genético e implementaciones auxiliares para crear juegos de datos y analizar los resultados. En la Sección 5 se presentan las pruebas realizadas y las evaluaciones de los resultados obtenidos. En las Secciones 6 y 7 se presentan nuestras conclusiones y posibles trabajos que se pueden realizar a futuro, respectivamente.

2. Conceptos de problemas de inventario

Como se mencionó anteriormente, el objetivo de este trabajo es atacar el problema de control de inventario con remanufacturación, clasificación y venta de retornos y disposición final.

A continuación, se presentan conceptos básicos de problemas de inventarios, por más detalle ver el Anexo 1 de Estado del Arte.

Se define como **inventario** al conjunto de productos (o artículos) que se deben mantener en algún lugar por un período de tiempo determinado y de los cuales se espera obtener algún beneficio.

Dado este concepto, podemos decir que el **control de inventario** es el conjunto de técnicas que ayudan a determinar cuándo y cuánto adquirir de un determinado artículo, para poder satisfacer los requerimientos de demanda sobre el mismo.

Un **problema de inventario** está dado cuando se desea gestionar un inventario, pero existen ciertas limitaciones e incertidumbres que obligan a llevar a la práctica un control de inventario. Estas limitaciones son, entre otras, físicas o de producción, mientras que las incertidumbres están asociadas con la venta y demanda.

La **remanufacturación** se define como la recuperación de los productos devueltos. La misma comprende las tareas de desmontaje, limpieza, pruebas, reemplazo, reparación parcial y reensamblaje, restaurando el producto a su calidad original.

La **clasificación de retornos** consiste en separar los retornos que llegan a la fábrica de acuerdo al nivel de desgaste o daño que tengan. Algo a destacar es que, a mayor desgaste, su costo unitario de remanufacturación es mayor y su costo de inventario es menor.

La **disposición final** es el proceso de desechar los retornos que no serán remanufacturados dado su nivel de calidad.

Los modelos de inventarios se pueden clasificar de acuerdo a diferentes criterios. A su vez estos criterios se pueden combinar entre sí.

- Según la cantidad de artículos, se pueden clasificar en modelos de inventario de **un artículo** o de **varios artículos**.
- Según el comportamiento de la demanda se pueden clasificar en **demanda determinística**: cuando se conoce o se asume conocida la demanda de un artículo en un período, o en **demanda estocástica**: cuando la demanda se comporta como una variable aleatoria con distribución conocida.
- En el caso de modelos de inventario para varios artículos, la demanda de cada uno de ellos puede ser considerada **independiente**, si no hay

relación entre los artículos, o **dependiente**, si la demanda de un artículo afecta la de otro.
- Por revisión del nivel de inventario se pueden clasificar en modelos de **revisión continua**: donde el nivel de inventario se puede determinar en cualquier momento, o en modelos de **revisión periódica**: donde el nivel de inventario se revisa cada cierto tiempo o en un momento particular en cada período.
- Según el horizonte de planeación, el modelo de inventarios se puede clasificar dentro de un **horizonte finito**, en caso que la cantidad de períodos sea limitada, o infinito si no.

A continuación, se definen algunos de los principales componentes del modelo de inventario, y en particular los vinculados al caso con opciones de retornos.

- **Tiempo de Entrega**: es el tiempo transcurrido entre una solicitud para reabastecer el inventario y el momento en que llegan los insumos. Dicho tiempo puede ser modelado como determinístico cuando es conocido (caso más utilizado en los modelos), o estocástico cuando se calcula en función de una distribución de probabilidad. Generalmente y para simplificar los problemas, este tiempo se supone cero (entrega instantánea).
- **Descuentos por cantidad**: por lo general, cuanto mayor sea el pedido realizado a un proveedor, menor será el precio unitario de un artículo. Una consecuencia de esto es que aumente el costo de mantener en inventario.
- **Costo de producción**: es el costo asociado a la fabricación de un artículo.
- **Costos de (mantener en) inventario**: es el costo asociado a almacenar en inventario un artículo, por un lapso de tiempo, hasta que el mismo sea vendido, utilizado o desechado.
- **Costo por faltante (o de penalización)**: es el costo que se produce cuando la demanda no es satisfecha en un período, es decir, la cantidad de artículos que hay en inventario no es suficiente para satisfacer la demanda. Pueden existir casos en los cuales esto no signifique un problema y se puedan satisfacer los requerimientos con la producción de períodos futuros. En caso de permitir faltantes, se consideran dos casos:
  - Nivel de inventario positivo: el nivel de inventario es siempre mayor o igual a cero. En caso de no cubrir un pedido con el nivel de inventario disponible, se realiza una producción urgente o se da la venta por pérdida.

    - Nivel de inventario negativo: si no se logra cubrir un pedido con el nivel de inventario actual, la demanda no se pierde y queda pendiente para ser satisfecha en un período posterior (backlogging).
- **Costo de remanufacturación**: es el costo asociado a la remanufacturación de un retorno. Este costo puede variar dependiendo de la calidad del mismo.
- **Costo de disposición final**: es el costo asociado a la eliminación de un retorno que no será remanufacturado.
- **Costos de configuración**: es el costo asociado a la preparación de las tareas de producción, remanufacturación o disposición final.
- **Valor de rescate**: es la ganancia asociada a la venta de un retorno que no ha sido remanufacturado.

3. Definición y modelado del problema ELSRC

En esta sección se presentan las características principales de los trabajos utilizados como base para la construcción del modelo matemático del problema. Se describen las características del ELSRC junto a su modelo matemático.

3.1. Base para la construcción del modelo

A continuación, se presenta parte de la literatura consultada que sirvió como base para la construcción del modelo matemático del problema ELSRC. Se puede consultar el Anexo 1 de Estado del Arte para obtener mayor detalle de toda la literatura revisada.

3.1.1. Problemas con remanufacturación y disposición final

El Problema de Tamaño de Lote Económico con Remanufacturación (ELSR) extiende el problema ELSP considerando remanufacturación de retornos. El objetivo es determinar cuánto producir y/o remanufacturar en cada período para cubrir la demanda a tiempo, minimizando los costos involucrados. Las características de este modelo son:

- un solo artículo
- demanda y cantidad de retornos conocida pudiendo variar en cada período
- revisión periódica o continua del nivel de inventario
- asume que la demanda es satisfecha al inicio de cada período
- la capacidad de producción y almacenamiento se supone ilimitada
- no se permite backlogging
- cada período cuenta con costos de producción y remanufacturación, costos de inventario, y costos de configuración.

En Yang et al. (2005) [27] se presenta el problema de ELSR donde se trabaja con costos cóncavos, y la demanda puede ser satisfecha con artículos nuevos o remanufacturados. Incluye disposición final, es decir, que los retornos que no son remanufacturados serán desechados. Presenta un algoritmo de programación dinámica para la resolución del problema y demuestran que el problema de costos cóncavos es NP-hard, incluso cuando todas las funciones de costo son estacionarias. Además, se desarrolla una heurística en tiempo polinomial de tiempo $O(T^4)$ para la resolución del problema con funciones de costos cóncavas.

Piñeyro & Viera (2009) [19] fue otro de los trabajos consultados para la formulación del problema estudiado ya que presenta un modelo ELSR con producción y disposición final. Este es un problema NP-hard. Maneja costos separados de producción, remanufacturación y disposición final, además de

costos de almacenamiento para productos listos (manufacturados y remanufacturados) y retornos. Plantea un conjunto de políticas (que se centran en la actividad de remanufactura) para resolver éste tipo de modelo, que son eficaces respecto al costo y al tiempo. Los problemas de producción y disposición final pueden ser resueltos de manera óptima por separado en $O(T^2)$ usando el algoritmo de Wagner-Whitin. La solución óptima para el ELSR se reduce entonces a encontrar el plan de remanufactura que permite determinar la producción óptima y el plan de disposición final.

3.1.2. ELSR con diferentes niveles de calidad

Ferguson et al. (2006) [10] presenta el problema ELSR con clasificación y venta de retornos. Esta clasificación consiste en determinar el estado en el que los retornos llegan a la fábrica, dado que los más deteriorados tendrán un costo de remanufacturación mayor que los menos deteriorados [18]. Esto introduce nuevos elementos al problema: costo fijo de clasificación, diferentes inventarios de retornos y costos de remanufacturación según las diferentes calidades. Se trabaja con una heurística de no inventario, cuyo resultado se compara contra la solución óptima. Dicha heurística asume que, en cada período, la cantidad de retornos es lo suficientemente alta como para no tener la necesidad de manufacturar, ya que la demanda se satisface solo con elementos remanufacturados.

La investigación numérica realizada en éste trabajo, muestra que un modelo con clasificación de retornos presenta una mejora con respecto al costo total del 11% (en promedio) frente a un sistema sin clasificación de retornos.

3.2.Características particulares del problema

El ELSRC hace referencia al problema de control de inventario con producción, remanufacturación y venta de retornos clasificados de acuerdo a su estado, así como la disposición final de los mismos.

La propuesta de modelo para el ELSRC presentada más adelante en esta sección se basó fuertemente en aquellos presentados en Piñeyro & Viera (2009) [19] y Ferguson et al. (2006) [10]. Del modelo presentado en Piñeyro & Viera (2009) se consideraron las actividades de remanufacturación y producción para satisfacer la demanda en cada período, además de la disposición final para los retornos que no serán remanufacturados. Del modelo presentado en Ferguson et al. (2006) se consideró la clasificación y venta de retornos según su calidad, planteado mediante una cantidad prefijada y finita de niveles, en donde a mayor nivel de calidad, menor costo de remanufacturación y mayor valor de rescate (valor o ganancia que se obtiene a partir de la venta). Cabe destacar que en la revisión de la literatura, no se

encontró ningún trabajo sobre el ELSR con clasificación de retornos como el que se considera para este proyecto.

Se supondrá que los retornos ya han sido inspeccionados y clasificados en uno de los posibles niveles de calidad prefijados, manejando un inventario de retornos por cada nivel de calidad. Estos retornos podrán ser remanufacturados o vendidos. Además, se considera un inventario independiente para los retornos a disponer finalmente sin distinguir ningún nivel de calidad entre ellos.

En la Figura 1 se representa el flujo de productos para el sistema de inventarios a considerar.

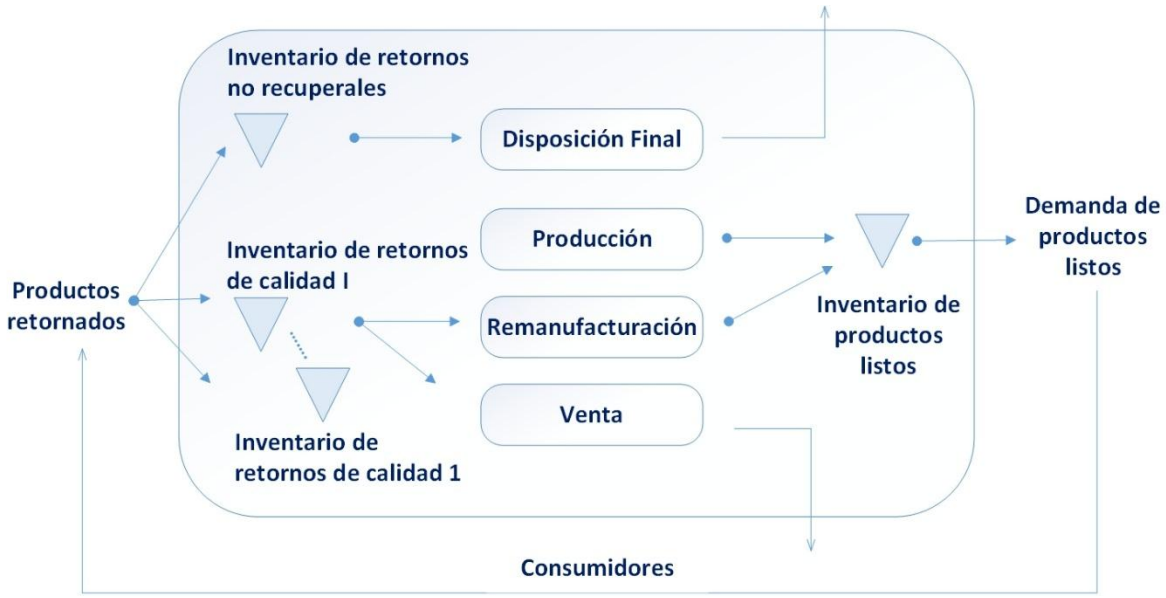


Figura 1: Esquema de flujo del Sistema

El problema contempla tres tipos diferentes de inventarios, según el estado de los productos. Los productos retornados podrán ir al inventario de retornos no recuperables (que eventualmente serán desechados) o al inventario de retornos a remanufacturar o vender, según sea su nivel de calidad. También se maneja un inventario de productos listos que contiene a los productos nuevos y remanufacturados.

El costo de mantener en inventario dependerá de la calidad de los retornos. Un retorno de peor calidad va a tener un costo de inventario menor, ya que se asume que los retornos de menor calidad son de menor valor y por lo tanto pueden requerir de menos cuidados, o hasta lugares más precarios para ser guardados, lo cual se traduce a costos de inventario inferiores.

Por otro lado, el valor de rescate al vender un retorno remanufacturable será positivo, ya que se asume que implica una ganancia para el fabricante.

Se consideran para el problema ELSRC las siguientes características:

- un solo artículo
- demanda determinística (conocida para cada período)
- revisión periódica del nivel de inventario
- horizonte de planeación finito
- actividades de producción, remanufacturación, venta y disposición final
- costos unitarios de producción, remanufacturación, venta y disposición final, diferentes por período y teniendo en cuenta la calidad para venta y remanufacturación
- costos de preparación para producción, remanufacturación y disposición final, diferentes por período y teniendo en cuenta la calidad para la remanufacturación
- costos unitarios de mantener en inventario para los retornos según su calidad, para los productos listos y para los retornos a disponer finalmente
- valor de venta unitario de los retornos remanufacturables (valor de rescate)
- niveles de inventarios iniciales iguales a cero
- tiempos de entrega cero
- capacidad de inventario y de producción infinita

No se considera:

- backlogging
- costos de penalización
- descuentos por cantidad

3.3.Modelo matemático

A continuación, se describen los componentes del modelo matemático para el ELSRC.

Parámetros:

- T: horizonte de planeación
- I: cantidad de niveles de calidad para los retornos
- D_t: demanda en el período t
- D_{jk}: demanda acumulada entre los períodos j y k
- $U_{i,t}$: cantidad de productos de calidad i que retornan en el período t
- $U_{ij,k}$: cantidad acumulada de retornos de calidad i entre los períodos j y k
- Q_t: cantidad de retornos no recuperables en el período t
- Q_{jk}: cantidad acumulada de retornos no recuperables entre los períodos j y k

- $s_{i,t}$: valor unitario de venta de un retorno de calidad *i* en el período t
- K_t^p: costo de preparación para producción en el período t
- $K_{i,t}^r$: costo de preparación para remanufacturación de retornos de calidad *i* en el período t
- K_t^d: costo de preparación para disposición final de retornos en el período t
- c_t^p: costo unitario de producir en el período t
- $c_{i,t}^r$: costo unitario de remanufacturar un retorno de calidad *i* en el período t
- c_t^d: costo unitario de disposición final de un retorno en el período t
- $h_{i,t}^u$: costo unitario de mantener en inventario un retorno de calidad *i* en el período t
- h_t^q: costo unitario de mantener en inventario un retorno no recuperable en el período t
- h_t^s: costo unitario de mantener en inventario un producto remanufacturado o nuevo en el período t
- $M \geq max\{D_{1T}, U_{1,1T}, U_{2,1T}, \dots, U_{I,1T}, Q_{1T}\}$

Variables de decisión:

- p_t: cantidad de productos nuevos a producir en el período t
- $r_{i,t}$: cantidad de retornos de calidad *i* a remanufacturar en el período t
- d_t: cantidad de retornos a disponer finalmente en el período t
- $v_{i,t}$: cantidad de retornos de calidad *i* que son vendidos en el período t
- $y_{i,t}^r$: nivel de inventario de productos a remanufacturar de calidad *i* en el período t
- y_t^q: nivel de inventario de productos a desechar en el período t
- y_t^s: nivel de inventario de productos listos en el período t
- $\delta_t^p = 1$ si en el período t la cantidad a producir es positiva, 0 en caso contrario
- $\delta_{i,t}^r = 1$ si en el período t la cantidad a remanufacturar de retornos de calidad *i* es positiva, 0 en caso contrario
- $\delta_t^d = 1$ si en el período t la cantidad a disponer finalmente es positiva, 0 en caso contrario

A continuación, se presenta el modelo matemático para el ELSRC:

$$min \sum_{t=1}^{T} \{ \sum_{i=1}^{I} \{K_{i,t}^r \delta_{i,t}^r + c_{i,t}^r r_{i,t} - s_{i,t} v_{i,t} + h_{i,t}^u y_{i,t}^r\} + K_t^p \delta_t^p + c_t^p p_t + K_t^d \delta_t^d + c_t^d d_t + h_t^q y_t^q + h_t^s y_t^s\} \quad (1)$$

s.a.

$$y_t^s = y_{t-1}^s + p_t + \sum_{i=1}^{I} r_{i,t} - D_t \qquad \forall\, t = 1, \dots, T \qquad (2)$$

$$y_{i,t}^r = y_{i,t-1}^r + U_{i,t} - r_{i,t} - v_{i,t} \qquad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \qquad (3)$$

$$y_t^q = y_{t-1}^q - d_t + Q_t \qquad \forall\, t = 1, \dots, T \qquad (4)$$

$$M\delta_t^p \geq p_t \qquad \forall\, t = 1, \dots, T \qquad (5)$$

$$M\delta_{i,t}^r \geq r_{i,t} \qquad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \qquad (6)$$

$$M\delta_t^d \geq d_t \qquad \forall\, t = 1, \dots, T \qquad (7)$$

$$y_0^s = y_{i,0}^r = y_0^q = 0 \qquad \forall\, i = 1, \dots, I \qquad (8)$$

$$\delta_t^p, \delta_{i,t}^r, \delta_t^d \in \{0, 1\} \qquad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \qquad (9)$$

$$p_t, r_{i,t}, d_t, v_{i,t}, y_{i,t}^r, y_t^s, y_t^q \geq 0 \qquad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \qquad (10)$$

Las siguientes observaciones se desprenden de la formulación (1)-(10):

- La disposición final puede ser considerada como un problema de optimización independiente. Esto se debe a que los retornos que entran para disponer finalmente, no se relacionan con los retornos que entran para ser remanufacturados o vendidos, manejando un inventario propio. La ecuación (1) se puede descomponer entonces en dos términos a optimizar que representan los dos problemas independientes:

$$\begin{aligned} &min \sum_{t=1}^{T} \{ \sum_{i=1}^{I} \{K_{i,t}^r \delta_{i,t}^r + c_{i,t}^r r_{i,t} - s_{i,t} v_{i,t} + h_{i,t}^u y_{i,t}^r\} + \\ &K_t^p \delta_t^p + c_t^p p_t + h_t^s y_t^s\} + min \sum_{t=1}^{T} \{K_t^d \delta_t^d + c_t^d d_t + h_t^q y_t^q\} \end{aligned} \qquad (11)$$

- Dado que la venta es un ingreso, se representa como un costo negativo.
- Los costos de preparación aplican a la realización o no de la actividad correspondiente. Mientras que los costos unitarios se aplican a la cantidad a realizar de esa actividad.
- Si la ganancia generada por la venta fuera lo suficientemente grande para competir con costos de preparación y costos unitarios, entonces sería más atractivo vender los retornos en lugar de remanufacturarlos.
- Si los costos de preparación son lo suficientemente grandes en comparación a los costos unitarios y de inventario, entonces será más atractivo realizar en menos períodos las actividades de producción y remanufacturación, pero en mayor cantidad para cumplir con la demanda.

El problema ELSRC formulado anteriormente es NP-hard, por ser una extensión del problema ELSR planteado en Piñeyro & Viera (2009), donde se muestra que el mismo es NP-hard. Se puede ver que el ELSR es un caso particular del ELSRC tomando los siguientes supuestos para el ELSRC:

- un solo nivel de calidad para los retornos

- venta cero en todos los períodos
- un solo ingreso de retornos a remanufacturar o disponer
- un solo inventario para retornos a remanufacturar o disponer

Partiendo entonces de las ecuaciones (1) a (10), considerando un solo nivel de calidad para los retornos y venta cero, el modelo matemático del ELSRC queda como sigue:

$$min \sum_{t=1}^{T} \{ K_t^r \delta_t^r + c_t^r r_t + h_t^u y_t^r + K_t^p \delta_t^p + c_t^p p_t + K_t^d \delta_t^d + c_t^d d_t + h_t^q y_t^q + h_t^s y_t^s \} \quad (12)$$

s.a.

$$y_t^s = y_{t-1}^s + p_t + r_t - D_t \qquad \forall\, t = 1, \dots, T \qquad (13)$$
$$y_t^r = y_{t-1}^r + U_t - r_t \qquad \forall\, t = 1, \dots, T \qquad (14)$$
$$y_t^q = y_{t-1}^q + Q_t - d_t \qquad \forall\, t = 1, \dots, T \qquad (15)$$
$$M\delta_t^p \geq p_t \qquad \forall\, t = 1, \dots, T \qquad (16)$$
$$M\delta_t^r \geq r_t \qquad \forall\, t = 1, \dots, T \qquad (17)$$
$$M\delta_t^d \geq d_t \qquad \forall\, t = 1, \dots, T \qquad (18)$$
$$y_0^s = y_0^r = y_0^q = 0 \qquad (19)$$
$$\delta_t^p, \delta_t^r, \delta_t^d \in \{0, 1\} \qquad \forall\, t = 1, \dots, T \qquad (20)$$
$$p_t, r_t, d_t, y_t^r, y_t^s, y_t^q \geq 0 \qquad \forall\, t = 1, \dots, T \qquad (21)$$

Luego, considerando un solo ingreso de retornos por período,

$$R_t = Q_t + U_t \qquad (22)$$

contando con un solo inventario de retornos,

$$y_t'^r = y_t^q + y_t^r \qquad (24)$$

donde el costo de mantener en ese inventario se calcula como

$$h_t'^r y_t'^r = h_t^q y_t^q + h_t^u y_t^r \qquad (25)$$

se llega al modelo matemático del ELSR presentado en Piñeyro & Viera (2009) que como ya se mencionó es un problema NP-hard.

$$min \sum_{t=1}^{T} \{ K_t^r \delta_t^r + c_t^r r_t + K_t^p \delta_t^p + c_t^p p_t + K_t^d \delta_t^d + c_t^d d_t + h_t'^r y_t'^r + h_t^s y_t^s \} \quad (26)$$

s.a.

$$y_t^s = y_{t-1}^s + p_t + r_t - D_t \qquad \forall\, t = 1, \dots, T \qquad (27)$$
$$y_t'^r = y_{t-1}'^r + R_t - r_t - d_t \qquad \forall\, t = 1, \dots, T \qquad (28)$$
$$M\delta_t^p \geq p_t \qquad \forall\, t = 1, \dots, T \qquad (29)$$
$$M\delta_t^r \geq r_t \qquad \forall\, t = 1, \dots, T \qquad (30)$$
$$M\delta_t^d \geq d_t \qquad \forall\, t = 1, \dots, T \qquad (31)$$
$$y_0^s = y_0'^r = 0 \qquad (32)$$
$$\delta_t^p, \delta_t^r, \delta_t^d \in \{0, 1\} \qquad \forall\, t = 1, \dots, T \qquad (33)$$
$$p_t, r_t, d_t, y_t'^r, y_t^s \geq 0 \qquad \forall\, t = 1, \dots, T \qquad (34)$$

Por lo tanto, al ser el ELSRC una extensión del ELSR, también es un problema NP-hard. Esto indica que, para instancias de gran tamaño, es poco probable que sea posible obtener en un tiempo razonable la solución óptima mediante un procedimiento computacional exacto.

4. Procedimientos de Resolución

Se realizaron dos implementaciones de procedimientos de resolución para el problema ELSRC. Una con GLPK para el modelo formulado en (1)-(10), y un procedimiento basado en la metaheurística de Algoritmos Genéticos. Más adelante los resultados de ambas serán comparados. A continuación, se describen cada uno de ellos.

4.1.GLPK

GLPK (GNU Linear Programming Kit) es un paquete de software destinado a la resolución de problemas de gran escala de programación lineal (LP), y programación entera mixta (MIP) entre otros [22].

Este paquete contiene un conjunto de rutinas escritas en ANSI C, que incluye componentes como métodos simplex, traductor para GNU MathProg, y solvers de LP/MIP.

En el contexto de este trabajo, GLPK se utilizó para obtener una solución óptima o cercana, de las diferentes instancias de datos, que se utilizaron como punto de comparación a la hora de medir la calidad del algoritmo genético desarrollado.

La codificación del modelo consta generalmente de dos archivos:

- archivo *.mod* (escrito en el lenguaje MathProg) que contiene el modelo, es decir, la definición de los conjuntos, las variables y parámetros, la función objetivo y sus restricciones. El archivo con la definición del modelo (ELSRC.mod) se incluye en el Anexo 2.
- archivo *.dat* en dónde se asignan valores a los parámetros del modelo. En el Anexo 2 se muestra un ejemplo de este archivo de entrada.

Dado que la definición del modelo se encuentra separada del archivo de datos, el mismo puede ser utilizado y ejecutado con diferentes datos de entrada.

GLPK es ejecutado desde línea de comandos. Se debe indicar el nombre del archivo que contiene el modelo, el nombre del archivo que contiene los datos y el nombre del archivo en el que se deberá guardar la solución encontrada. Opcionalmente se puede indicar mediante un parámetro el tiempo máximo (en segundos) de ejecución para obtener la solución. Este parámetro es útil para aquellos problemas de complejidad NP-hard, en donde las ejecuciones pueden demorar horas o incluso días, si el tamaño de las instancias es grande. Por este motivo, es necesario acotar el tiempo de ejecución, aunque GLPK no llegue a

una solución óptima. La ejecución vuelca datos en consola y los mismos pueden ser direccionados a un archivo de salida.

4.2.Algoritmo Genético

Para problemas de mediano y gran porte, encontrar una solución de forma exacta puede llevar a tiempos extensos de procesamiento dado que este tipo de problemas son NP-hard. Una metaheurística nos permite afrontar el problema de cierta forma acortando considerablemente los tiempos de procesamiento y aun así llegar a una solución que se aproxima mucho a la solución óptima.

Los Algoritmos Genéticos (AGs) son un tipo de metaheurística que pueden utilizarse para resolver problemas de búsqueda y optimización. Se basan en la teoría de Darwin (1859) de evolución de las especies. La idea de AGs fue introducida por Holland en 1975 y se apoya en la idea de supervivencia y evolución de soluciones (individuos) a un problema difícil de resolver. Al igual que las especies en la teoría de Darwin, las soluciones forman una población que evoluciona de cierta manera tal que sólo las mejores soluciones van "sobreviviendo" en cada generación [2].

El motivo principal por el cual se decidió utilizar la metaheurística de algoritmos genéticos fue que los integrantes del proyecto contaban con conocimientos previos en dicho enfoque. Por otra parte, entre la literatura consultada para el proyecto, casi no se encontraron trabajos sobre el problema del tamaño del lote con remanufacturación en los que se empleara esta metaheurística, aunque si para el problema clásico sin retornos [12].

La Figura 2 muestra un diagrama de flujo del algoritmo genético en términos generales. El algoritmo itera sobre la cantidad de ejecuciones independientes que se deben realizar. Primero se cargan los parámetros del algoritmo, luego, para cada ejecución, se genera una población inicial de individuos, y se hace evolucionar la población a través de distintas generaciones. La evolución de la población de una generación a otra, consiste en evaluar a cada individuo asignándole un valor que determinará la aptitud del mismo para sobrevivir. Luego se aplica un conjunto de operadores sobre esta población que determinarán una nueva población. Los operadores consisten en: seleccionar un conjunto de individuos aptos para reproducirse, cruzar algunos de los individuos seleccionados creando dos nuevos individuos a partir de dos progenitores y mutar algunos de los individuos para introducir diversidad en la población. Esto determina un nuevo conjunto de individuos (población) que conforma la siguiente generación.

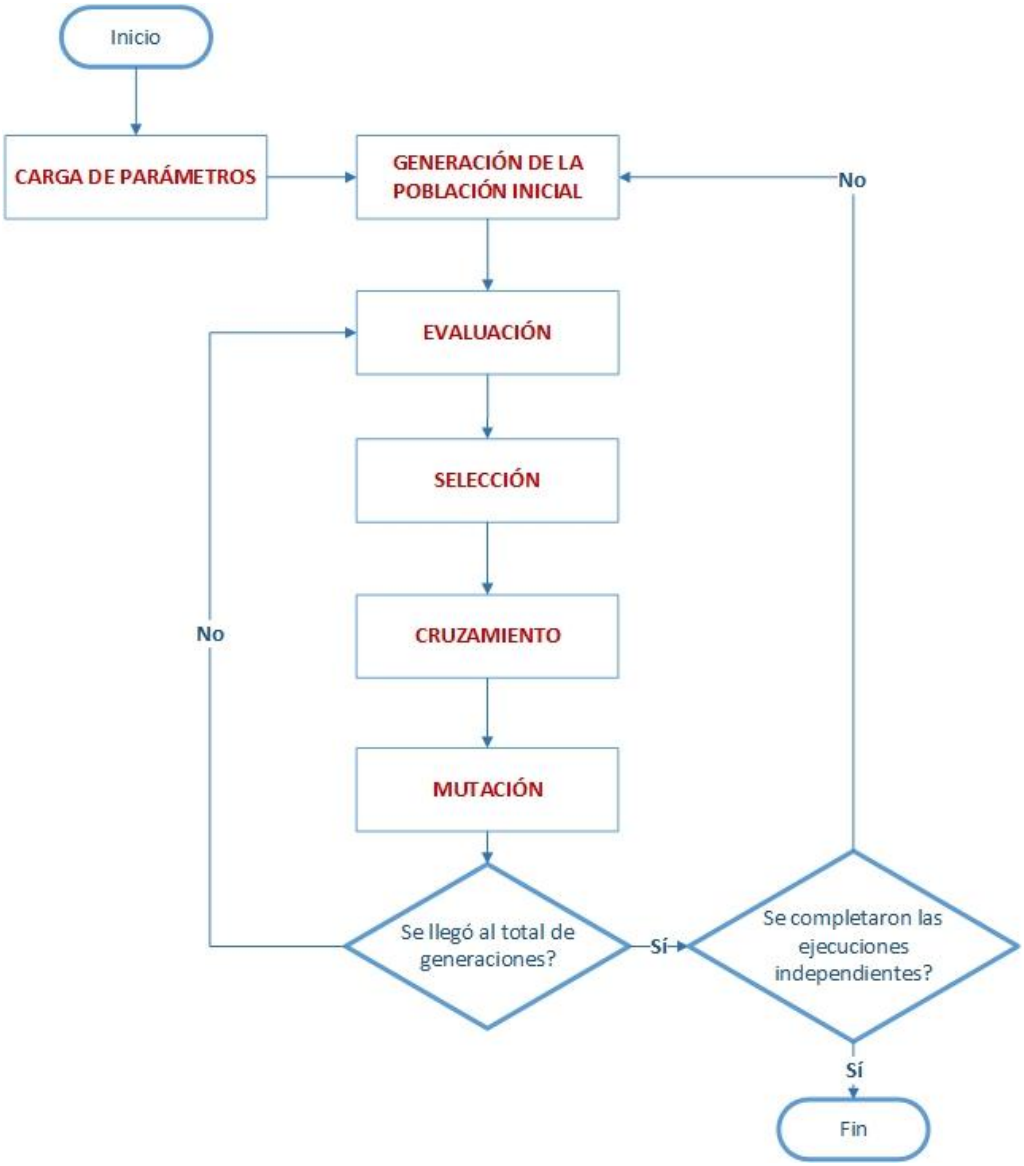


Figura 2: Diagrama de flujo del AG

Los componentes básicos de un AG son: la representación de las soluciones, un procedimiento para crear una población inicial, una función de evaluación llamada función de fitness (la cual se utiliza para evaluar a cada individuo de la población permitiendo clasificarlo en términos de su aptitud), un conjunto de operadores de evolución que modifican la composición de los individuos a través de las generaciones y una configuración paramétrica [1].

A continuación, se detallan las características de estos componentes para la implementación del procedimiento de solución para el ELSRC.

4.2.1. Representación de una solución

La representación de la solución es a través de un vector de enteros. Su tamaño varía según si el problema se modela con disposición final, siendo

igual a *2*T * (1 + I)*, o sin disposición final, igual a *T * (1 +2*I)*. Donde *T* es la cantidad de períodos e *I* son los distintos niveles de calidad de los retornos.

Se describe el significado de las posiciones en este vector:

- Los primeros *T* elementos del vector corresponden a las cantidades a producir en cada período (variables $p_1,\ldots, p_t$ del modelo matemático).
- Los siguientes *T*I* elementos del vector corresponden a las cantidades de retornos a remanufacturar de cada calidad para cada período (variables $r_{1,1}, r_{1,2}, \ldots, r_{1,t}, r_{2,1}, r_{2,2}, \ldots, r_{2,t}, \ldots. r_{i,t}$ del modelo matemático).
- Los siguientes *T*I* elementos del vector corresponden a las cantidades de retornos a vender de cada calidad para cada período *(*variables $v_{1,1}, v_{1,2}, \ldots, v_{1,t}, v_{2,1}, v_{2,2}, \ldots, v_{2,t}, \ldots.. v_{i,t}$ del modelo matemático).
- En caso que se modele disposición final, los últimos *T* elementos corresponden a las cantidades de retornos a desechar en cada período (variables $d_1, \ldots, d_t$ del modelo matemático).

A modo de ejemplo, en un problema con *T=3* e *I=2* donde se maneja disposición final, la Figura 3 muestra la estructura del vector que representa a los individuos donde:

- p_t contiene la cantidad a producir en el período *T.*
- $r_{i,t}$ contiene la cantidad a remanufacturar de un retorno de calidad *i* en el período *t.*
- $v_{i,t}$ contiene la cantidad a vender de un retorno de calidad *i* en el período *t.*
- d_t contiene la cantidad a disponer en el período *T.*

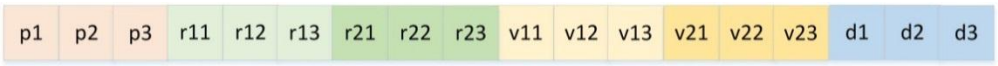


Figura 3: Vector que representa un individuo

4.2.1. Procedimiento para crear la población inicial

En el inicio del desarrollo del AG se trabajó con una población inicial generada aleatoriamente. Esto trajo como consecuencia que se iniciara con un espacio de soluciones de muy mala calidad con respecto al objetivo buscado de minimizar los costos, donde la mayoría de los individuos eran no factibles, haciendo mucho más costosa la evolución del algoritmo y no permitiendo llegar a resultados factibles en algunos casos.

Una solución se considera factible si para las cantidades a producir, remanufacturar, vender y desechar, los niveles de los distintos inventarios son mayores o iguales a cero para todos los períodos.

Se definió entonces un método para generar la población inicial, aplicando la aleatoriedad sobre las actividades. Luego calculando las cantidades para cada actividad, es decir, cuánto se produce, se remanufactura, se vende o se desecha en cada período para cada calidad, asegurando que los individuos sean factibles.

El método se aplica al crear cada individuo que va a formar parte de la población inicial. Consiste en crear el vector que representa al individuo con valores binarios elegidos de forma aleatoria. Estos valores indican si la actividad a la que corresponde la posición en el vector, se realizará o no. Luego en base a este vector, se calculan las cantidades de cada actividad y se actualiza el mismo con estas cantidades mayores o iguales a cero. El método creado para calcular estas cantidades se explica en profundidad más adelante en la Sección 4.2.7.

Durante el transcurso del proyecto se observó, para los diferentes tipos de juegos de datos, que los resultados devueltos por GLPK tenían cierta densidad en cuanto a la cantidad de períodos donde se realiza cada actividad con respecto al total de períodos para esa actividad. Por ejemplo, en el caso de juegos de datos con costos variables, se realizaba la actividad de producción en aproximadamente un 30% de los períodos. De esto surgió la idea de elegir diferentes proporciones de 0s y 1s por actividad en los vectores que representan los individuos de la población inicial, para juegos de datos con costos constantes o variables. Con esto se logró aproximar la cantidad de actividades a realizar en los individuos de la población inicial a la solución de GLPK, obteniendo mejores resultados en menor tiempo.

4.2.2. Función de Fitness

La función de fitness determina los individuos candidatos a sobrevivir de una población en una cierta generación. Guía el mecanismo de búsqueda a través del operador de selección. Dado que para el ELSR el modelo matemático consiste en minimizar la suma de los cotos totales, la función de fitness se definirá entonces como el opuesto de la función a minimizar:

$$F = -\sum_{t=1}^{T}\{\sum_{i=1}^{I}\{K_{i,t}^{r}\delta_{i,t}^{r} + c_{i,t}^{r}r_{i,t} - s_{i,t}v_{i,t} + h_{i,t}^{u}y_{i,t}^{r}\} + K_{t}^{p}\delta_{t}^{p} + c_{t}^{p}p_{t} + K_{t}^{d}\delta_{t}^{d} + c_{t}^{d}d_{t} + h_{t}^{q}y_{t}^{q} + h_{t}^{s}y_{t}^{s}\} \quad (35)$$

En el caso de que no se modele la disposición final, los términos $K_t^d\delta_t^d, c_t^d d_t, h_t^q y_t^q$ no se consideran.

4.2.3. Operadores

En cada generación se aplican los operadores evolutivos. Primero se seleccionan los padres de acuerdo a su valor de fitness, se cruzan generando descendientes, estos descendientes son mutados de forma aleatoria y pasan a formar parte de la población, remplazando algunos individuos de la generación anterior.

4.2.3.1. Operador de Selección

La selección dirige el algoritmo hacia la exploración de secciones del espacio de búsqueda que pueden ser prometedoras, es decir donde se puede encontrar soluciones de buena calidad, cercanas a la óptima.

Como operador de selección se utilizó la Selección por Torneo, donde se selecciona el mejor individuo de un conjunto aleatorio de k individuos por torneo (en este caso $k = 2$). En otras palabras, se eligen al azar una cantidad k de individuos padres del cual se selecciona el mejor de ellos (de acuerdo a su valor de fitness) para reproducirse. El ganador de ese torneo, pasará a la siguiente generación.

Como implementación de esta selección se tomó la provista por ECJ [4], biblioteca que se mencionará en la Sección 4.2.5.

4.2.3.2. Operador de Cruzamiento

El cruzamiento busca garantizar la explotación de regiones buenas o prometedoras del espacio de búsqueda.

Como operador de cruzamiento se implementó un método que primero convierte dos vectores de enteros (que representan dos individuos) a dos vectores de actividad con valores binarios, que indican si la actividad (de producción, remanufacturación, venta o disposición final) se realiza o no. Cada posición del vector que vale 0 se mantiene en ese valor, y en otro caso pasa a valer 1. Luego sobre estos vectores de actividad se aplica un cruzamiento de dos puntos obteniendo dos vectores que son cruza de los anteriores (ver Figura 4).

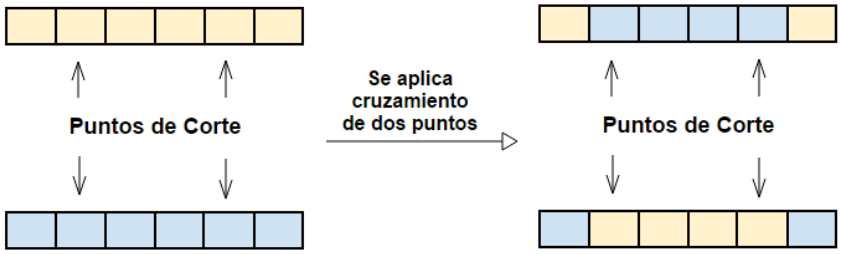


Figura 4: Cruzamiento de dos puntos

A estos vectores cruzados se le calculan las cantidades a realizar de cada actividad, pasando nuevamente a ser vectores de enteros donde cada posición indica la cantidad a realizar de cada actividad. Este método es el mismo que el utilizado al crear la población inicial y se explica en profundidad en la Sección 4.2.7.

4.2.3.3. Operador de Mutación

La mutación aporta a la exploración de diferentes regiones del espacio de búsqueda, introduciendo diversidad de forma aleatoria en los individuos de la población.

Como operador de mutación se implementó un método que primero decide, según una probabilidad de ajuste, entre dos opciones:

- Unificar en el individuo dos actividades de producción para evitar los costos de configuración en ambos períodos. Con una probabilidad de 0,5 se buscan en el individuo: dos períodos consecutivos para unificar o dos períodos seguidos, pero no consecutivos. En ambos casos se verifican los costos unitarios de producción y los costos de mantener en inventario, con el objetivo de que el ajuste no incremente los costos totales.
    - En el primero de los casos, cuando se encuentran dos períodos consecutivos en los que se realiza la actividad de producción, se pasa lo producido en el segundo período al primero, dejando en cero la actividad de producción del segundo período (ver Figura 5).

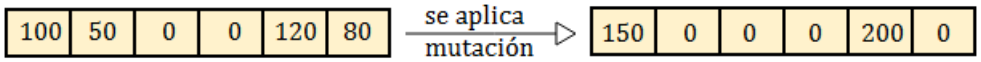


Figura 5: Mutación en períodos consecutivos de producción.

    - El segundo de los casos es igual al anterior, aunque los períodos no necesariamente son consecutivos (ver Figura 6).

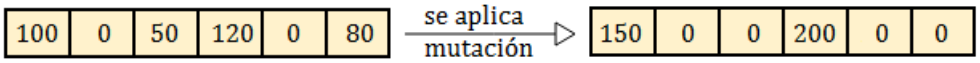


Figura 6: Mutación en períodos no consecutivos de producción.

- Transformar el vector que representa al individuo en un vector de actividades con valores binarios, donde 0 significa que la actividad (de producción, remanufacturación, venta o disposición final) no se realiza y 1 que sí se realiza. La transformación

aplicada a cada posición del vector que representa al individuo es como sigue:

- Si el valor es igual a 0
  - y si de acuerdo a la probabilidad de mutación toca mutar, se cambia el valor a 1
  - si no toca mutar, se deja el valor en 0
- Si el valor es distinto a 0
  - y si de acuerdo a la probabilidad de mutación toca mutar, se cambia el valor a 0
  - si no toca mutar, se cambia el valor a 1

Luego sobre este vector de actividades se aplica el método de cálculos de las cantidades a realizar de cada actividad, al igual que en el cruzamiento y en la generación de la población inicial.

4.2.4. Configuración paramétrica

Es posible configurar diferentes parámetros para la ejecución del algoritmo a través de un archivo de configuración o también a través de línea de comando.

Uno de los parámetros que se definen es el tamaño de la población (*pop.subpop.0.size)*, que se mantiene fijo durante la ejecución del algoritmo.

También se puede asignar por parámetro la probabilidad bajo la cual se aplica el operador de cruzamiento a un individuo (*pop.subpop.0.species.pipe.source.0.likelihood*), la probabilidad bajo la cual se aplica el operador de mutación a un elemento del vector que representa al individuo (*pop.subpop.0.species.pipe.mutation-prob*) y la probabilidad con la cual se realiza un ajuste en las actividades de producción que se explicó bajo el operador de mutación (*pop.subpop.0.species.pipe.probAjuste*).

Como criterio de parada se determina por parámetro la cantidad de generaciones que se quieren en la evolución del algoritmo.

También es posible indicar, a través de parámetros booleanos, si se debe tener en cuenta la disposición final o no (*conDisposicionFinal*), si el modelo maneja ventas (*conVenta*) y si maneja costos unitarios constantes o variables (*costosUnitariosFijos*).

En el Anexo 3 se brinda una descripción detallada de todos los parámetros útiles para el algoritmo desarrollado.

Más adelante en la Sección 5.2 se describen las configuraciones paramétricas elegidas luego de un análisis empírico llevado a cabo para determinarlas.

4.2.5. Biblioteca elegida para la implementación

Se utilizó ECJ [4], biblioteca de computación evolutiva escrita en Java, de código abierto, que soporta algoritmos genéticos y otras metaheurísticas.

Otra opción que se manejó, fue la de utilizar la biblioteca Mallba [5] escrita en C++. Por experiencia previa resultó más amigable trabajar en Java con las funcionalidades provistas por ECJ.

Esta biblioteca requiere:

- La implementación de una clase que inicialice el problema obteniendo los datos de entrada del algoritmo, y que implemente la función de evaluación de un individuo asignándole su valor de fitness.
- Además, requiere la creación de un archivo de parámetros con determinado formato, que permite configuraciones como las mencionadas en la Sección 4.2.4 junto a otras configuraciones de métodos o representaciones a utilizar.
- La implementación de una clase de estadísticas que muestre el resultado del problema de la forma que se desee.

Todo lo demás es provisto por esta biblioteca, pudiendo extenderse y/o modificarse.

4.2.6. Arquitectura de la solución

En la Figura 7 se muestra un diagrama de clases con la arquitectura de la solución. Vale aclarar que la biblioteca es mucho más amplia, pero en esta figura solo se muestran las principales clases que aplican al algoritmo.

En naranja se resaltan las clases que debieron ser agregados a la estructura de ECJ junto al archivo de parámetros del problema. En verde se indican las clases ya existentes en la biblioteca pero que tuvieron alguna modificación. Y en celeste se indican clases ya existentes y no modificadas.

La clase Evolve carga los parámetros (desde el archivo de parámetros) al inicio de la ejecución y crea el elemento del nivel más alto de la evolución, EvolutionState.

La clase Initializer se encarga de inicializar la población al comienzo de la ejecución y la clase Breeder se encarga de actualizarla a través de las generaciones proveyendo nuevos individuos.

La clase IntegerVectorIndividual representa al individuo como vector de enteros.

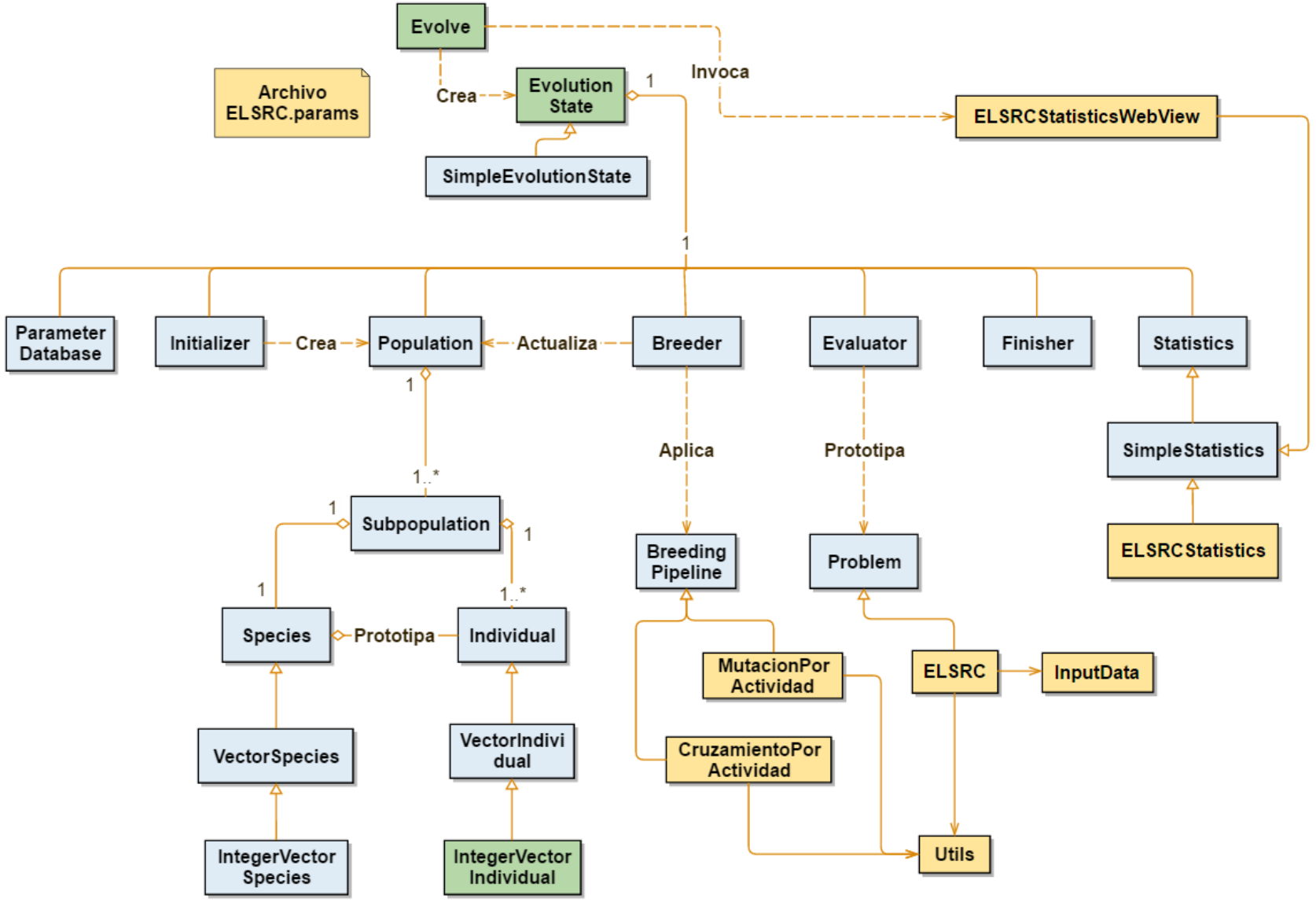


Figura 7: Arquitectura de AG

La clase Evaluator es la encargada del proceso de evaluación durante el curso de la evolución. En cada generación, los individuos de la población son evaluados contra el problema implementado en la clase ELSRC, el cual cuenta con datos de entrada como demanda por períodos, costos de configuración, etc. Durante la evaluación, a cada individuo se le asigna un valor de fitness calculado como se explicó en la Sección 4.2.2 el cual determina qué individuo es mejor que otro.

Se desarrollaron dos clases de estadísticas diferentes. ELSRCStatistics que genera un archivo de salida por cada ejecución independiente del algoritmo, con el resultado del mejor individuo en esa ejecución. Y la clase ELSRCStatisticsWebView que genera una vista más amigable en formato *.html* con los resultados para el mejor individuo elegido entre los mejores de cada ejecución independiente. Este último es invocado desde la clase Evolve que es quien itera en las distintas ejecuciones independientes. Sobre estas salidas del algoritmo se explican más detalles en la Sección 4.2.8.

4.2.7. Técnicas aplicadas

Se utilizaron dos técnicas para asegurar que la evolución de los individuos a lo largo de la ejecución del algoritmo, fuera hacia una región de soluciones factibles del ELSRC.

La primera de las técnicas se aplica durante la evaluación de un individuo y consiste en penalizar el valor de su fitness, en caso de que algún nivel de inventario sea negativo para algún período. Como valor de fitness se le asigna al individuo un valor grande negativo, de forma de eliminarlo del conjunto de soluciones al momento de la selección por torneo.

La otra técnica se aplica durante la mutación y el cruzamiento de individuos, y consiste en calcular las cantidades a realizar de cada actividad por período, asegurando que los niveles de inventario sean siempre positivos, dejando a los individuos dentro de la región factible. Posteriormente esta técnica fue aplicada durante la inicialización de la población, dejando redundante la primera de las técnicas descriptas ya que es seguro que los individuos generados siempre están dentro de la región factible.

El método de cálculo de cantidades se aplica sobre vectores binarios donde cada posición corresponde a la realización o no de una actividad en un período determinado, transformándolo en un vector de enteros, que representa al individuo, donde cada posición indica cuánto se realiza de la actividad correspondiente en el período correspondiente. La Figura 8 muestra un ejemplo de un vector de entrada y el de salida luego de aplicar el método.

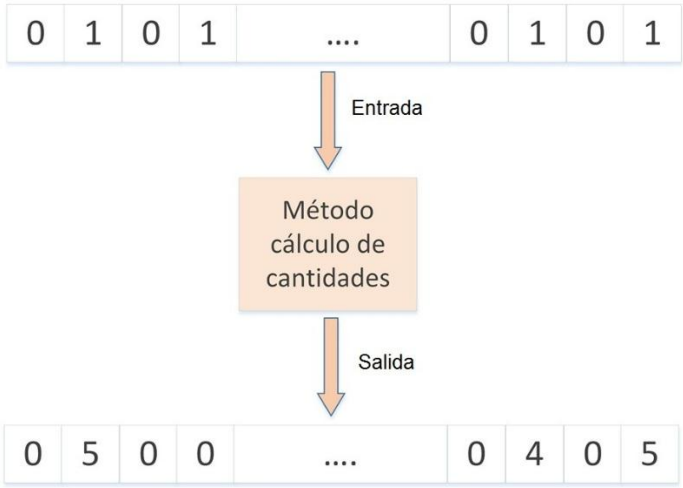


Figura 8: Cálculo de cantidades

Se definen tres políticas sobre la remanufacturación de retornos de diferentes calidades:

1. Remanufacturar primero los retornos de mejor calidad.
2. Remanufacturar primero los retornos de peor calidad.
3. Remanufacturar los retornos de diferentes calidades de forma aleatoria.

El método comienza eligiendo una de estas tres políticas a aplicar de forma aleatoria, dando más peso a las políticas 1 y 2 que dieron mejores resultados al aplicarlas en la mayoría de los casos. Luego se va iterando en los períodos calculando para cada uno de ellos las cantidades a remanufacturar, vender, producir y desechar.

Solo se remanufactura, se vende o se desecha, en aquellos períodos donde el vector binario indica que se realiza la actividad y si se cuenta con material para hacerlo. En el caso de la producción, se produce si el vector binario indica producción en ese período o si la remanufacturación no alcanza a cubrir la demanda del período.

Para la cantidad a remanufacturar se tiene en cuenta la política elegida antes, y si se está trabajando con costos unitarios constantes o variables.

En el caso de costos unitarios constantes, se remanufactura una sola calidad (siguiendo el orden en las calidades según la política elegida) sólo si cubre la demanda en ese período, y de ser así, se remanufacturan todos los retornos disponibles, es decir lo que había en inventario más los que entraron en ese período para esa calidad. De lo contrario no se remanufactura ningún retorno de esa calidad y se hace el mismo chequeo con la siguiente calidad. Luego de definir cuánto remanufacturar de una calidad, se define cuánto vender de los retornos de esa misma calidad; y se calcula como un valor aleatorio entre cero y la cantidad de retornos

disponibles menos lo que se decidió remanufacturar, para esa calidad en ese período.

En el caso de costos unitarios variables, se puede llegar a remanufacturar más de una calidad, siguiendo el orden en las calidades según la política elegida. A diferencia del caso de costos unitarios contantes, donde solo se remanufactura de una calidad si la misma alcanza a cubrir la demanda en ese período. Esta cantidad a remanufacturar por período, se calcula como el mínimo entre los retornos disponibles (retornos entrantes más nivel de inventario de retornos del período anterior) y la demanda acumulada entre el período actual y un período anterior al próximo período donde el vector binario indica remanufacturación. La venta se calcula de la misma forma que en el caso de costos unitarios constantes.

La distinción entre los cálculos de las cantidades a remanufacturar por costos unitarios constantes o variables, surgió del análisis de las soluciones obtenidas por GLPK y de las pruebas realizadas sobre el AG, para los distintos juegos de datos generados.

En el caso de costos unitarios constantes se observó que las mejores soluciones obtenidas por GLPK solo realizaban una actividad por período, remanufacturación o producción. Y en la mayoría de los casos solo se remanufacturaba una de las calidades.

En el caso de costos unitarios variables esta política no daba buenos resultados en el AG, y además en las soluciones obtenidas por GLPK, se realizaba más de una actividad por período. Por lo tanto, se permitió realizar más de una actividad y remanufacturar más de una calidad, por período.

La cantidad a producir se calcula con un valor aleatorio entre un mínimo y un máximo.

El mínimo se calcula como: lo demandado en ese período, menos lo remanufacturado en ese período, menos el nivel de inventario de productos listos en el período anterior. Y el máximo se calcula como: la demanda acumulada desde el período actual, hasta un período anterior al próximo período de producción.

En el ejemplo de la Tabla 1 se quiere calcular la cantidad a producir en el período 4.

T	**¿Producir?**	**Demanda**	**Cantidad Remanufacturados**	**Inventario de P. Listos**	**Cantidad a Producir**
⋮	⋮	⋮	⋮	12	⋮
4	si	34	5		X
5	no	9	2		
6	no	14	7		

7	no	21	12		
8	si	6	1		
⋮	⋮	⋮	⋮	⋮	⋮

Tabla 1: Ejemplo de planificación de producción.

Esta cantidad *X* se calcula como un número aleatorio entre:

- La demanda en el período 4, menos lo remanufacturado en el período 4, menos el inventario de productos listos en el período 3. Esto es lo mismo que: 34 – 5 – 12 = 17.
- La demanda acumulada entre los períodos 4 y 7 (período anterior al próximo donde hay producción): 34 + 9 + 14 + 21 = 78.

Por lo tanto, se tiene que la cantidad a producir en el período 4 es un número aleatorio entre el mínimo y máximo calculados.

La cantidad a desechar se obtiene de un valor aleatorio entre cero y la cantidad en inventario de desechos más lo que entró para desechar en ese período.

Una vez que se llega al último período en la iteración, se ajustan las cantidades a producir de modo que el nivel de inventario de productos listos en este último período sea cero. El excedente de producción se va restando a la actividad de producción de forma iterativa en los períodos en orden descendente, hasta que el mismo sea cero.

En la Figura 9 se muestra un diagrama de flujo, sobre el método de cálculo de cantidades para cada una de las actividades diseñado para el algoritmo.

4.2.8. Entrada y salida del algoritmo

Como entrada el algoritmo recibe un archivo de datos con todos los parámetros del problema a resolver, tales como demanda por períodos, retornos por calidad por períodos, costos de configuración de producción por períodos, etc. Este archivo es el mismo utilizado como entrada para resolver el problema con GLPK y se explica en el Anexo 2.

Como se mencionó antes en la Sección 4.2.6, el algoritmo cuenta con dos salidas posibles. La primera en implementarse consiste en un archivo de texto con extensión *.out.stat* generado por cada ejecución independiente del algoritmo. Este archivo contiene información no solo de la solución al problema, es decir de los valores de las variables del problema, sino también de otros datos que permiten evaluar el algoritmo, tales como tiempo del algoritmo, tiempo en hallar la mejor solución, cómo fue evolucionando la solución, en qué número de generaciones se halló el mejor

individuo, etc. Un ejemplo de este archivo de salida puede encontrarse en el Anexo 3.

La segunda en implementarse consiste en un archivo *.html* que contiene para la mejor solución encontrada de entre todas las ejecuciones independientes, las cantidades a producir, remanufacturar, vender y disponer, y los niveles de inventarios de productos listos, de retornos por calidad y de retornos a disponer, calculados para cada período. También el costo total obtenido para esa solución y el tiempo en milisegundos que tomó el algoritmo. El Anexo 3 muestra esta salida.

4.2.9. Ejecución del algoritmo

La ejecución del algoritmo puede ser por línea de comandos, o a través de un ejecutable generado por un formulario web. Por una descripción detallada de ambos ver el Anexo 3.

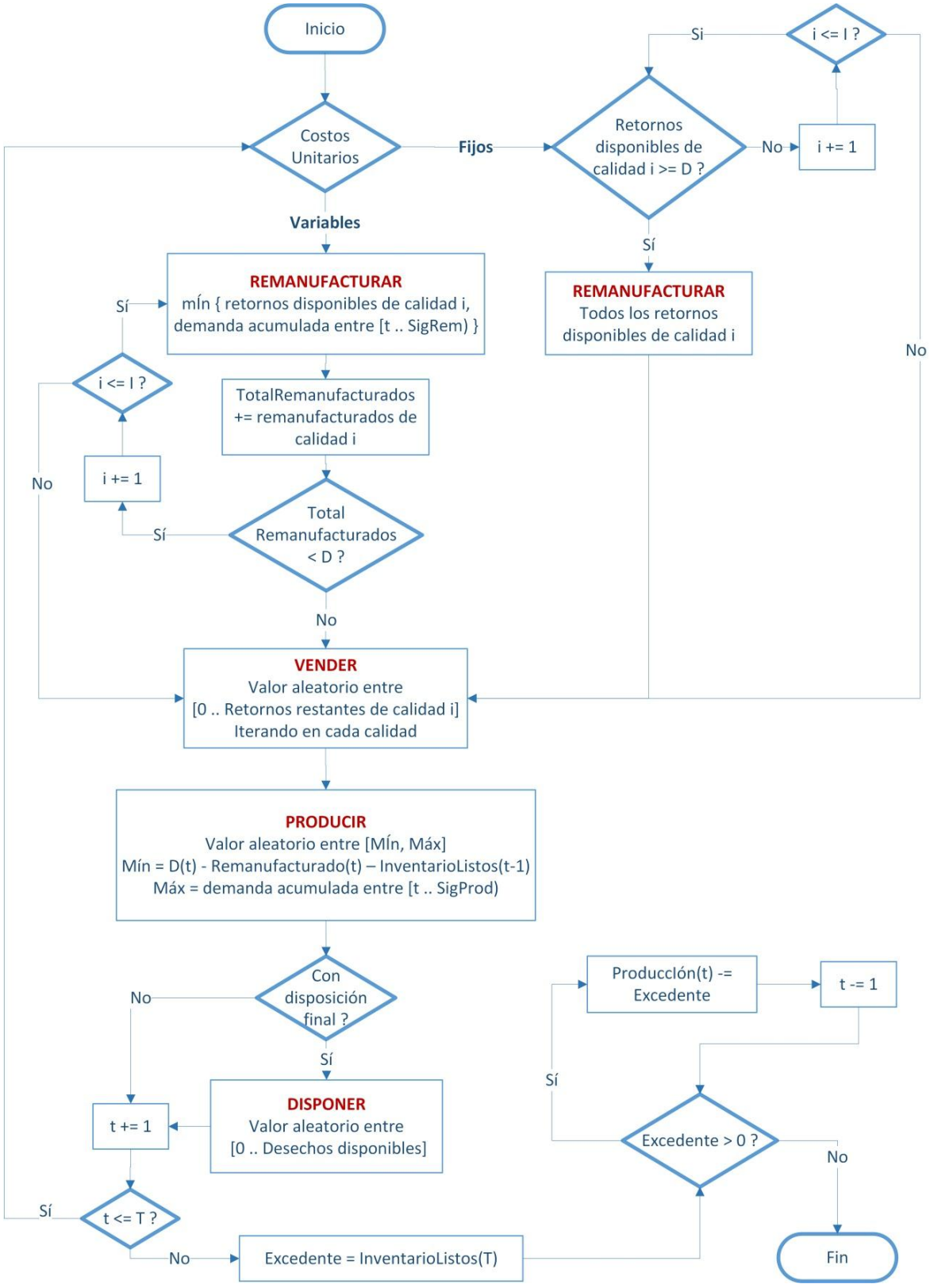


Figura 9: Diagrama de flujo, cálculo de cantidades por actividad.

4.3.Herramientas Auxiliares

Se realizaron dos implementaciones auxiliares al problema a resolver. La primera permite generar archivos de entrada utilizados por GLPK y el algoritmo genético implementado; la segunda facilita el análisis de las soluciones obtenidas.

A continuación, se explican dichas implementaciones.

4.3.1. Generador de datos

Se implementó una aplicación para generar archivos de entrada del problema a resolver, la cual fue desarrollada en C#.

La misma permite especificar los diferentes parámetros que maneja el problema, permitiendo guardarlos en un archivo de configuración con extensión *.gdd*. A partir de esta configuración, permite generar tantos archivos de datos diferentes como se desee, con extensión *.dat*.

Los parámetros que se deben ingresar para definir una configuración son:

- cantidad de períodos
- cantidad de calidades de retornos
- si modela disposición final o no
- cantidad de archivos de datos independientes a generar
- cómo generar las cantidades para demanda, retornos, desechos
- cómo generar los costos de configuración para manufacturación, remanufacturación y disposición final
- cómo generar los costos unitarios para artículos manufacturados y remanufacturados, retornos desechados y vendidos
- cómo generar los costos de inventario de artículos listos, de retornos y de desechos

Por más detalles sobre la implementación, consultar el manual del generador de datos, Anexo 4.

4.3.2. Análisis de resultados

Esta herramienta permite extraer, organizar y estudiar los resultados de las pruebas realizadas. La misma fue desarrollada en lenguaje C#.

Como resultado se obtiene un archivo en formato *.csv* comparando las salidas del algoritmo genético y de GLPK. El archivo contiene los mejores tiempos y costos, la evolución del fitness, porcentajes de distancia del algoritmo genético con respecto a GLPK, entre otros.

Por más detalles sobre la implementación, consultar el manual del analizador de resultados, Anexo 5.

5. Pruebas realizadas

En esta sección se describe la forma en que fueron generados los juegos de datos y las configuraciones paramétricas elegidas para ejecutar GLPK y el algoritmo genético.

Por las características estocásticas del algoritmo genético, cada archivo de entrada fue ejecutado 30 veces. Para las ejecuciones realizadas con GLPK, se especificó en todos los casos que el tiempo máximo de ejecución fuera de 30 minutos.

5.1. Juegos de datos utilizados

Utilizando el generador de datos implementado descripto en la Sección 4.3.1, se armaron diferentes archivos de configuración (*.gdd*) teniendo en cuenta las características presentadas por los autores consultados durante la realización del estado del arte. A partir de los archivos de configuración, se generaron los juegos de datos utilizados para correr el modelo en GLPK y el algoritmo genético.

La principal diferencia entre los juegos de datos armados es el tamaño de las instancias y que algunos manejan costos unitarios constantes y otros costos unitarios variables.

Los trabajos seleccionados como base para armar los juegos de datos fueron: Sifaleras et al. (2015) [21], Schulz (2011) [20] y Yang et al. (2005) [27], dado que los mismos presentaban características similares al trabajo realizado y estaban disponibles los criterios que utilizaron en la selección de los valores utilizados en las pruebas.

Fue necesario realizar algunas adaptaciones, dado que los mismos no utilizaban más de una calidad. Los valores presentados en estos trabajos, fueron tomados como los valores de la calidad media de los juegos de datos construidos.

También se armaron juegos de datos propios con algunas de las características presentadas en el trabajo de Sifaleras et al. (2015).

5.1.1. Costos unitarios constantes

5.1.1.1. Instancias basadas en Sifaleras et al. (2015)

Se armaron 108 archivos de configuración con las características presentadas en el trabajo de Sifaleras et al. (2015). La cantidad de configuraciones se corresponde con la cantidad de casos de prueba que se presentan en dicho trabajo y fueron adaptados dado que el mismo maneja una sola calidad para los retornos.

Por cada archivo de configuración se armaron 2 archivos de datos, porque al manejar instancias grandes de 52 períodos, los tiempos de procesamiento eran demasiado extensos, contando entonces con un total de 216 archivos de entrada.

Los siguientes valores fueron utilizados para armar los datos de prueba:

- *52* períodos
- 3 calidades para la clasificación de los retornos
- La cantidad demandada (*D*) se generó como una distribución normal con media 100 y la desviación estándar tomó valores de 10 y 20.
- La cantidad de retornos (*U*) se generó como una distribución normal con media 17, donde la suma de la media de los retornos de cada calidad se tomó como el 50% de la media de la demanda. La desviación estándar de esta distribución normal, tomó valores de 2 y 3 que corresponden al 10% y 20% de la media de los retornos de cada calidad.
- El costo de configuración para la manufacturación (*Kp*) tomó los valores 200, 500 y 2000.
- El costo de configuración para la remanufacturación (*Kr*) tomó los valores 200, 500 y 2000 para la calidad 2. En los casos de 200 y 500, las calidades 1 y 3 tomaron valores de menos/más 100 respectivamente, y para el caso de 2000, las calidades 1 y 3 tomaron valores de menos/más 500 respectivamente.
- El costo de inventario para productos listos (*hs*) se fijó en 1.
- El costo de inventario para retornos (*hu*) tomó los valores 0.2, 0.5 y 0.8 para la calidad 2, teniendo una distancia de más/menos 1 para las calidades 1 y 3 respectivamente.
- Los costos unitarios de manufacturación y remanufacturación se fijaron en 0.
- No se manejó disposición final ni venta.

5.1.1.2. Instancias basadas en Schulz (2011)

Se armaron 162 archivos de configuración con las mismas características presentadas en el punto anterior y algunas adaptaciones. En éste caso también se manejaron 3 calidades y se realizó una variación en la cantidad de retornos que se tomó como el 30%, 50% y 70% de la media de la demanda.

Por cada archivo de configuración se armaron 10 archivos de datos, teniendo un total de 1620 archivos de entrada. A diferencia del caso anterior, se armaron más archivos de entrada porque al manejar

instancias pequeñas de 12 períodos, los tiempos de procesamiento eran menores.

Los valores que variaron fueron los siguientes:

- 12 períodos
- La cantidad de retornos (U) se generó como una distribución normal con media 10, 17 y 23, donde la suma de la media de los retornos de cada calidad, se tomó como el 30%, 50% y 70% de la media de la demanda (que vale 100), generando retornos bajos, medios o altos con respecto a la demanda. La desviación estándar de esta distribución normal, tomó valores de 1 y 2, que corresponden al 10% y 20% de la media de los retornos.

5.1.2. Costos unitarios variables

5.1.2.1. Instancias basadas en Yang et al. (2005)

Se armaron 6 archivos de configuración con las características presentadas en el trabajo de Yang et al. (2005). La cantidad de configuraciones se corresponde con la cantidad de períodos utilizados en dicho trabajo (10, 12, 14, 16, 18 y 20) y fueron adaptados dado que el mismo maneja una sola calidad para los retornos. Además, la disposición final se tomó como una calidad ficticia en el cálculo de datos de entrada. Y se ajustaron los costos unitarios, de inventario y de configuración multiplicándolos por un factor de 10, para trabajar con costos enteros.

Por cada archivo de configuración se armaron 5 archivos de datos, teniendo un total de 30 archivos de entrada.

Los siguientes valores fueron utilizados para armar los datos de prueba:

- 10, 12, 14, 16, 18 y 20 períodos
- 3 calidades para la clasificación de los retornos
- La cantidad demandada (D) se generó con una distribución de Poisson de lambda 10.
- La cantidad de retornos (U) y la cantidad a disponer finalmente (Q), se generaron con una distribución de Poisson de lambda 1,25. Se eligió este valor de lambda dado que Yang maneja una distribución con lambda 5 para los retornos, sin distinguir entre los que serán remanufacturados o desechados. En el ELSRC se distinguen los retornos a desechar del resto. Con el fin de aproximar la entrada de retornos a la distribución de Yang, para los retornos a desechar, se tomó una calidad ficticia, dividiendo el valor de lambda entre 4 (las 3 calidades más la ficticia).
- El costo de configuración para la manufacturación (Kp) se generó con una Distribución Uniforme con mínimo 100 y máximo 150.

- El costo de configuración para la remanufacturación (*Kr*) se generó con una Distribución Uniforme con mínimo 30 y máximo 36 para la calidad 2. Las calidades 1 y 3 tomaron valores de menos/más 7 respectivamente.
- El costo de configuración para la disposición final (*Kd*) se generó con una Distribución Uniforme con mínimo 10 y máximo 20.
- El costo unitario de manufacturación (*cp*) se generó con una Distribución Uniforme con mínimo 30 y máximo 50.
- El costo unitario de remanufacturación (*cr*) se generó con una Distribución Uniforme con mínimo 10 y máximo 13 para la calidad 2. Las calidades 1 y 3 tomaron valores de menos/más 3,5 respectivamente.
- El costo unitario de disponer finalmente (*cd*) se generó con una Distribución Uniforme con mínimo 5 y máximo 10.
- El costo de inventario para productos listos (*hs*) se generó con una Distribución Uniforme con mínimo 10 y máximo 20.
- El costo de inventario para retornos (*hu*) se generó con una Distribución Uniforme con mínimo 9 y máximo 10 para la calidad 2. Las calidades 1 y 3 tomaron valores de más/menos 2 respectivamente.
- El costo de inventario para disposición final (*hq*) se generó con una Distribución Uniforme con mínimo 3 y máximo 4.
- No se manejó venta.

5.1.2.2. Instancias propias (GLM)

Se armaron juegos de datos propios con algunas de las características presentadas en el trabajo de Sifaleras et al. (2015). Los archivos de configuración presentan las siguientes características:

- 12 períodos y 3 calidades para la clasificación de los retornos. Se diferenció entre casos de cantidad de retornos bajos, medios y altos, con respecto a la demanda.
- 24 períodos y 4 calidades para la clasificación de los retornos.
- 52 períodos y 3 calidades para la clasificación de los retornos.
- Para los casos de 24 y 52 períodos sólo se consideró una cantidad media de retornos con respecto a la demanda.
- Todos los casos manejan disposición final y venta de retornos.
- Por cada caso se generó 1 archivo de configuración (5 en total) y de cada uno se generaron 10 archivos de datos.

Los siguientes valores fueron utilizados para armar los datos de prueba:

- La cantidad demandada (D) se generó con una distribución normal de media 100 y desviación estándar 10.
- La cantidad de retornos (U) se generó con una distribución normal:
  - ✓ retornos bajos: media 3 y desviación estándar 1.
  - ✓ retornos medios: media 15 y desviación estándar 1.
  - ✓ retornos altos: media 25 y desviación estándar 1.
- La cantidad a disponer finalmente (Q) se generó con una distribución normal de media 5 y desviación estándar 1.
- El costo de configuración para la manufacturación (Kp) se generó con una Distribución Uniforme con mínimo 450 y máximo 550.
- El costo de configuración para la remanufacturación (Kr) se generó con una Distribución Uniforme con mínimo 50 y máximo 150 para la calidad 2. Las calidades 1 y 3 tomaron valores de menos/ más 100 respectivamente.
- El costo de configuración para la disposición final (Kd) se generó con una Distribución Uniforme con mínimo 20 y máximo 50.
- El costo unitario de manufacturación (cp) se generó con una Distribución Uniforme con mínimo 100 y máximo 200.
- El costo unitario de remanufacturación (cr) se generó con una Distribución Uniforme con mínimo 10 y máximo 20 para la calidad 2. Las calidades 1 y 3 tomaron valores de menos/más 10 respectivamente.
- El costo unitario de disponer finalmente (cd) se generó con una Distribución Uniforme con mínimo 5 y máximo 10.
- El costo unitario de vender un retorno que no se desecha ni remanufactura se generó con una Distribución Uniforme con mínimo 8 y máximo 10 para la calidad 2. Las calidades 1 y 3 tomaron valores de más/menos 2 respectivamente.
- El costo de inventario para productos listos (hs) se generó con una Distribución Uniforme con mínimo 10 y máximo 20.
- El costo de inventario para retornos (hu) se generó con una Distribución Uniforme con mínimo 3 y máximo 4 para la calidad 2. Las calidades 1 y 3 tomaron valores de más/menos 1 respectivamente.
- El costo de inventario para disposición final (hq) se generó con una Distribución Uniforme con mínimo 1 y máximo 2.

5.2.Evaluación de configuración paramétrica

Se armaron diferentes configuraciones paramétricas (8 en total) y se hicieron varias corridas del algoritmo genético para determinar qué

configuración paramétrica era la más adecuada según las características del problema: costos constantes o variables, muchos o pocos períodos, entre otros.

En una primera instancia se armaron 6 configuraciones diferentes, luego de realizar algunas ejecuciones y observando los resultados obtenidos, se agregaron las configuraciones 7 y 8. Los parámetros que variaron entre las diferentes configuraciones fueron la probabilidad de ajuste (probAjuste) y la probabilidad de mutación (mutation-prob), que tomaron los valores en la Tabla 2 a continuación:

Parámetros AG	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
pop.subpop.0.species.pipe.probAjuste	0.01	0.01	0.65	0.65	0.3	0.3	0.3	0.65
pop.subpop.0.species.pipe.mutation-prob	0.1	0.2	0.1	0.2	0.1	0.2	0.3	0.3

Tabla 2: Parámetros AG – configuraciones.

La cantidad de generaciones (generations), el tamaño de la población (pop.subpop.0.size) y la probabilidad de cruzamiento (pop.subpop.0.species.pipe.source.0.likelihood), se dejaron fijos y tomaron los valores especificados en la Tabla 3:

Parámetros AG	**Valor**
generations	10000
pop.subpop.0.size	500
pop.subpop.0.species.pipe.source.0.likelihood	0.5

Tabla 3: Parámetros AG.

A su vez, para cada una de las configuraciones anteriores y según el caso con el que se estuviera trabajando, variaron los parámetros presentados en la Tabla 4, dado que no todos manejan disposición final, venta y costos unitarios constantes:

	Sifaleras et al.	**Schulz**	**Yang et al.**	**GLM**
conDisposicionFinal	false	false	true	true
conVenta	false	false	false	true
costosUnitariosFijos	true	true	false	false

Tabla 4: Parámetros AG.

Por cada marco de referencia, se seleccionaron algunos archivos de datos para correr el algoritmo genético con las diferentes configuraciones. La cantidad seleccionada por marco de referencia varió dependiendo de la complejidad del problema. En todos los casos se seleccionaron archivos

que cumplían con las diferentes características mencionadas en la Sección 5.1.

Luego de realizar una primera ejecución para todos los datos, se analizaron los resultados obtenidos para determinar cuál era la mejor configuración según las características de cada caso.

Por otro lado, viendo en cada caso el tiempo en el que el algoritmo genético encontró la mejor solución y la cantidad de generaciones que le llevó, se ajustaron por cada autor la cantidad de generaciones a realizar, con el objetivo de reducir el tiempo de ejecución del algoritmo genético.

Para determinar la mejor configuración en cada caso, se analizaron los resultados obtenidos, teniendo en cuenta cuál brindaba el mejor resultado con respecto a la solución obtenida para los mismos archivos de entrada con GLPK. Para esto se definió la distancia (valor porcentual) entre el valor objetivo obtenido por el AG y el valor objetivo obtenida por GLPK como:

$$distancia = (costoMínAG - costoGLPK) * \frac{100}{costoGLPK} \qquad (36)$$

- ✓ Para los archivos de Sifaleras et al. (2015) y Schulz (2011) se consideraron buenas soluciones aquellas en las que la distancia no superaba el 5%. Nótese que son casos en dónde los costos unitarios son fijos y no se maneja venta ni disposición final.
- ✓ Cómo el resto de los casos (Yang et al. (2005) y GLM) son más complejos ya que tienen costos unitarios variables, manejan venta (ambos) y disposición final (GLM), se consideraron buenas soluciones aquellas en las que la distancia no superaba el 10%.

Las configuraciones seleccionadas para correr el algoritmo genético para cada autor fueron las indicadas en la Tabla 5 a continuación:

Autor	**Configuración**	**Generaciones**
Schulz	8	5000
Sifaleras et al.	8	8000
Yang et al.	6	8000
GLM - retornos altos	8	8000
GLM - retornos bajos	4	5000
GLM - retornos medios	5	8000

Tabla 5: Configuraciones seleccionadas para ejecutar el AG.

El Anexo 6 contiene las soluciones obtenidas para cada archivo de datos con las diferentes configuraciones.

5.3.Evaluación de resultados obtenidos

Luego de determinar qué configuración paramétrica era la más adecuada para ejecutar el algoritmo genético según las características de cada caso, se procedió a la ejecución de todos los juegos de datos generados (los mencionados en la Sección 5.1).

En el Anexo 7 de Resultados se pueden consultar los resultados obtenidos que se analizan a continuación.

5.3.1. Instancias basadas en Sifaleras et al. (2015)

Como se mencionó anteriormente, se crearon 216 archivos de entrada. Los mismos fueron utilizados para ejecutar el modelo especificado en GLPK y el algoritmo genético desarrollado.

Para ninguna de estas entradas GLPK alcanzó la solución óptima, consumiendo para todas ellas el tiempo máximo de ejecución predefinido de 30 minutos.

En el caso del AG el tiempo de cada ejecución fue restringido por la cantidad de generaciones (8000) y el tamaño de la población (500 individuos), tomando un promedio de 1 minuto y medio aproximadamente.

El AG mejoró las soluciones obtenidas por GLPK en el 82.4% de los casos, mientras que estuvo a una distancia del 5% en el restante 17.6% de los casos. No hubo casos donde el algoritmo genético igualara la solución de GLPK, ni casos donde el valor objetivo de la mejor solución obtenida por el AG fuera 5% mayor al valor objetivo de la solución obtenida por GLPK.

En la Figura 10 se muestran las comparaciones del resultado obtenido en GLPK con la solución mejor, promedio y peor (de las 30 ejecuciones) del AG para todos los casos. Se puede observar que el AG nunca obtuvo la misma solución que GLPK, pero si lo superó o se mantuvo dentro de un rango aceptable en su mayoría.

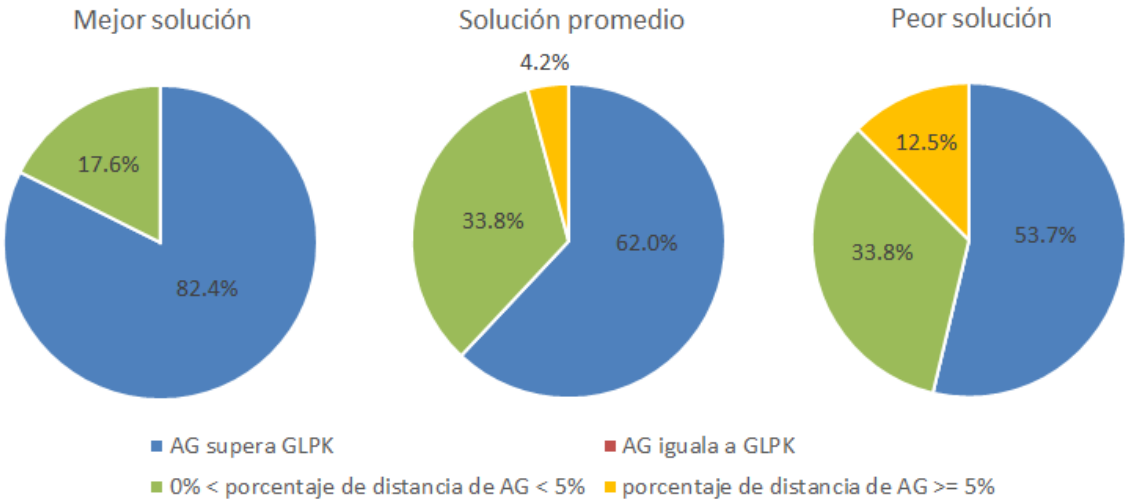


Figura 10: Comparación de mejor, promedio y peor solución de AG contra GLPK.

La gráfica de la Figura 11 compara los valores para la mejor solución obtenida por el AG contra los valores de GLPK, para una muestra de 27 juegos de datos tomados de entre los 216 generados. Esta muestra fue seleccionada simplemente con el objetivo de que abarcara diferentes combinaciones de los parámetros mencionados en la Sección 5.1.1.1.

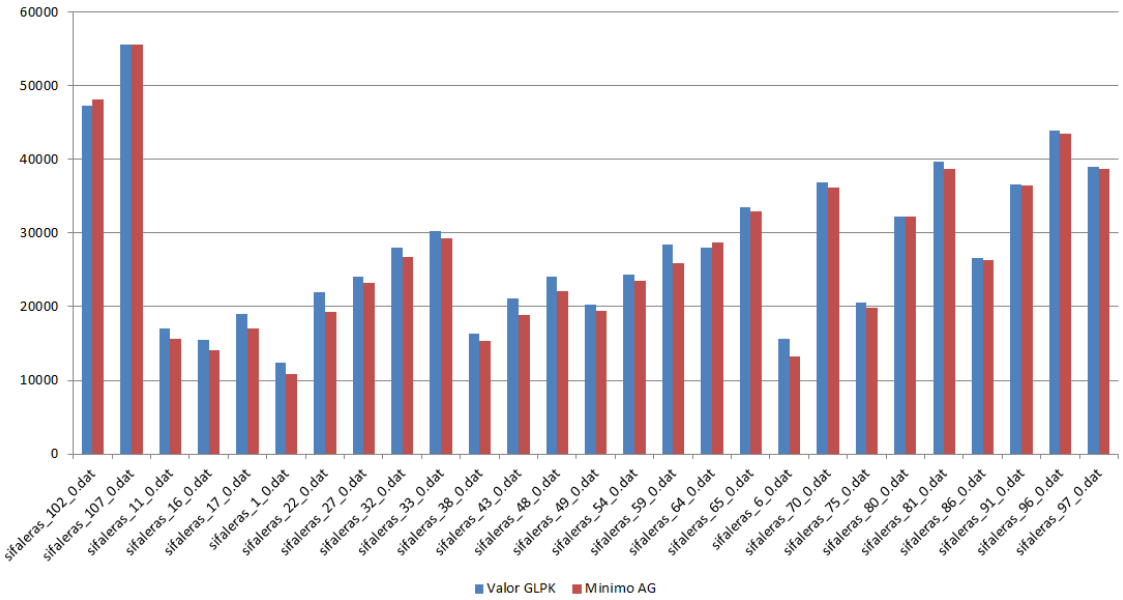


Figura 11: Comparación entre resultados de GLPK y AG en una muestra reducida.

Como se puede observar, en la mayoría de los casos el valor obtenido por el AG es menor al obtenido por GLPK y en un tiempo de ejecución sensiblemente menor.

La gráfica de la Figura 12 presenta para la muestra anterior el porcentaje de distancia. Como se puede observar, solo en 3 de estos casos el porcentaje es mayor a cero, pero no supera el 4%. Los casos donde el

porcentaje es negativo indican que el AG encontró una mejor solución que GLPK.

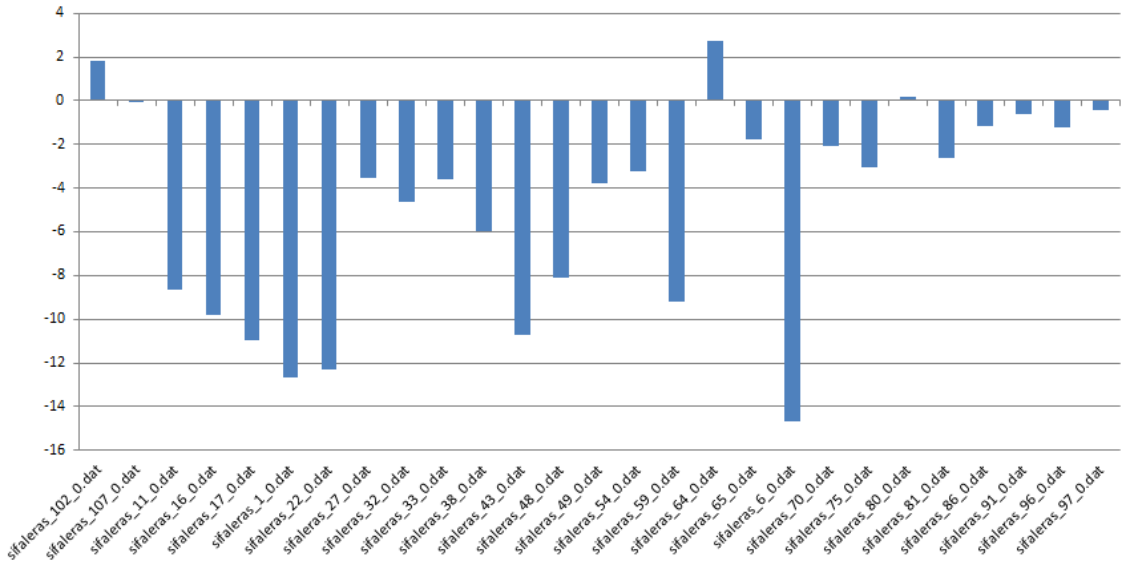


Figura 12: Porcentaje de distancia para una muestra reducida.

Durante los experimentos realizados también se registró la evolución del fitness a lo largo de las diferentes generaciones, con el objetivo de observar cómo se comportaba el AG. Esto permitió reducir la cantidad de generaciones en las ejecuciones asegurando obtener de igual manera una buena solución.

La Figura 13 muestra la evolución del fitness, a través de las generaciones, para el archivo de entrada "*Sifaleras_10_1.dat*". Se puede observar que el valor del fitness aumenta acorde van avanzando las generaciones. Los saltos representan una mejora en la solución obtenida hasta el momento.

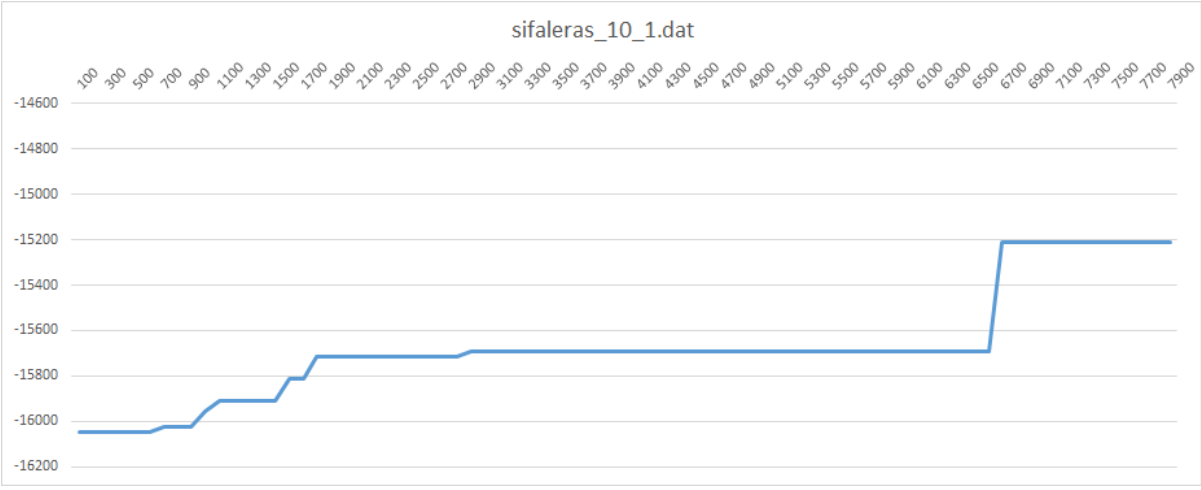


Figura 13: Evolución del fitness para el archivo "Sifaleras_10_1.dat"

La Figura 14 es otro ejemplo de la evolución del fitness, a través de las generaciones, para el archivo de entrada "*Sifaleras_90_0.dat*".

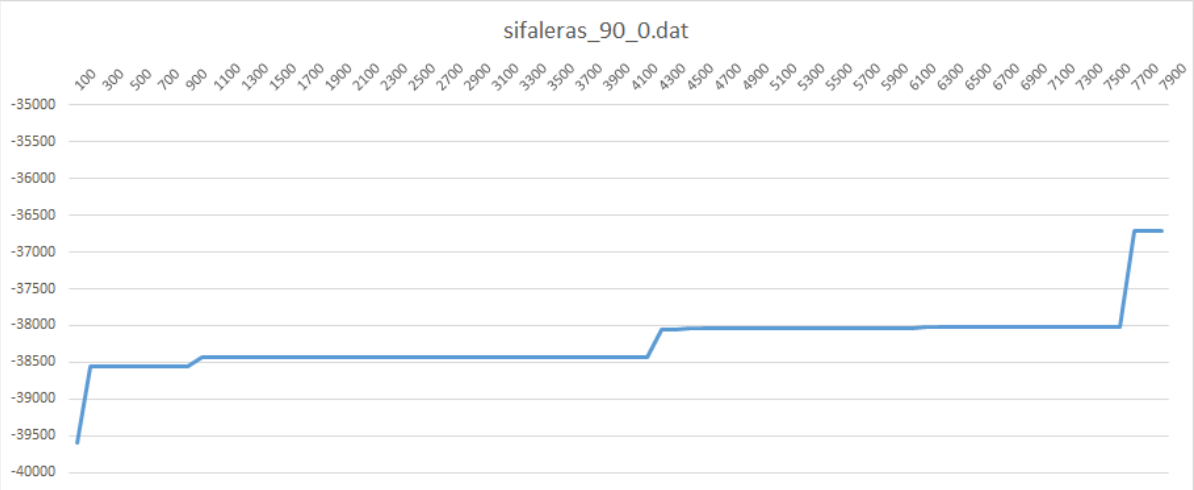


Figura 14: Evolución del fitness para el archivo "Sifaleras_90_0.dat"

5.3.2. Instancias basadas en Schulz (2011)

Para los 1620 archivos de entrada generados como se explicó antes, GLPK alcanzó la solución óptima en un 99% de los casos (1600), tomando para los mismos un tiempo promedio de 8 segundos.

En el caso del AG el tiempo de cada ejecución fue restringido por la cantidad de generaciones (5000) y el tamaño de la población (500 individuos), tomando un tiempo promedio de 15 segundos.

Analizando las mejores soluciones obtenidas, el AG superó a GLPK en un 10%, lo igualó en un 35.1%, estuvo a menos del 5% en un 53.9%, y entre el 5% y el 8.5% en el 1% de los casos restantes.

La Figura 15 muestra las comparaciones del resultado obtenido en GLPK con la solución mejor, promedio y peor (de las 30 ejecuciones) del AG para todos los juegos de datos. Se puede observar que el AG superó el porcentaje considerado como aceptable del 5% en menos de un 2% de los casos.

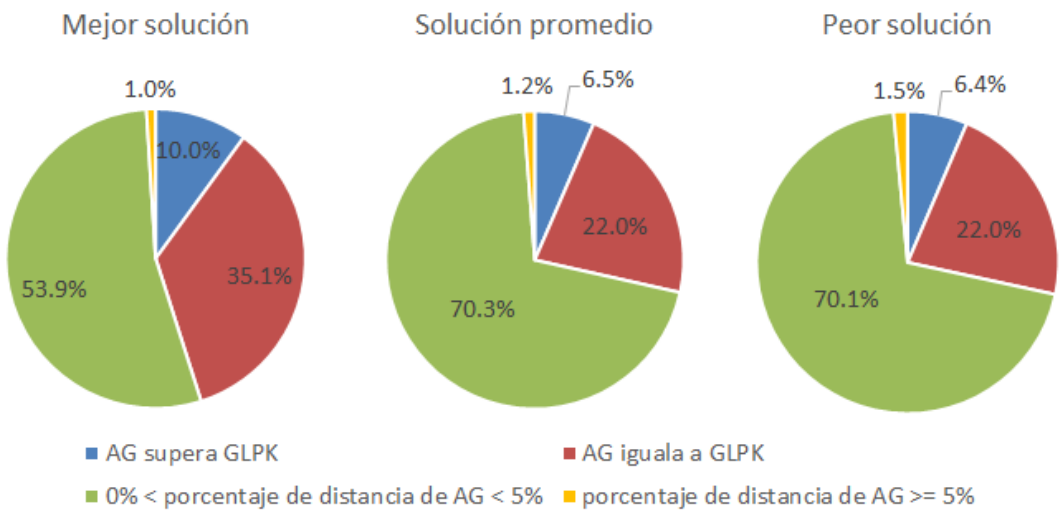


Figura 15: Comparación de mejor, promedio y peor solución de AG contra GLPK.

La gráfica de la Figura 16 muestra la comparación de los valores para la mejor solución obtenida por el AG contra la solución de GLPK, para una muestra de 27 juegos de datos tomados de entre los 1620 generados. Esta muestra fue seleccionada simplemente con el objetivo de que abarcara diferentes combinaciones de los parámetros mencionados en la Sección 5.1.1.2. Se observan valores muy cercanos en la mayoría de los casos.

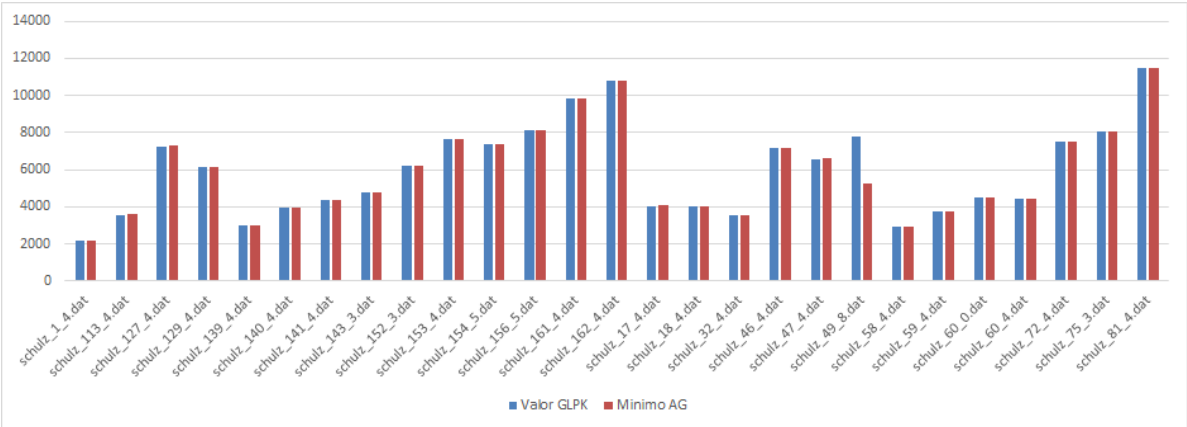


Figura 16: Comparación entre resultados de GLPK y AG en una muestra reducida.

La gráfica de la Figura 17 presenta para la misma muestra de datos, el porcentaje de distancia de la mejor solución obtenida por el AG respecto a GLPK. Donde se puede observar que las soluciones están a lo sumo a una distancia de 1.2%.

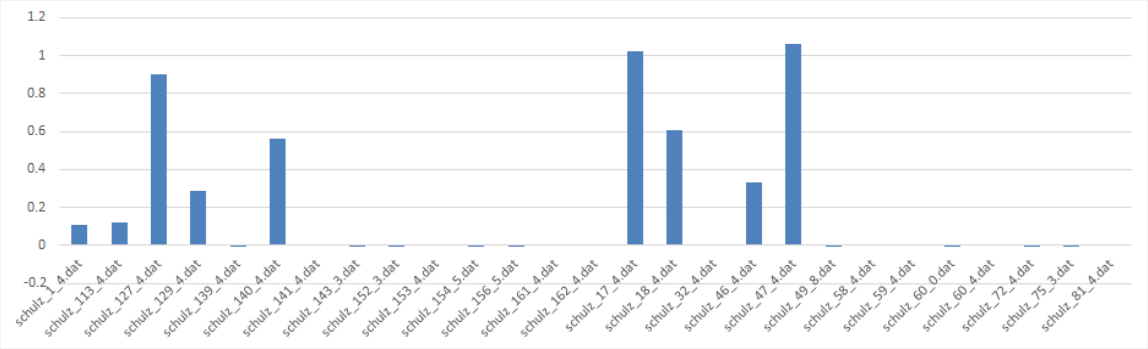


Figura 17: Porcentaje de distancia para una muestra reducida.

En la Tabla 6, se muestran los porcentajes donde el AG superó, igualó, se mantuvo por debajo del 5% aceptable, o superó este 5%, con respecto a GLPK, para los juegos de datos agrupados por retornos bajos, altos o medios con respecto a la demanda.

Retornos	Solución	AG supera GLPK	AG iguala a GLPK	0% < porcentaje de distancia de AG < 5%	porcentaje de distancia de AG >= 5%
Bajos	Mejor	0%	54.1%	45.9%	0%
	Promedio	0%	33.5%	66.5%	0%
	Peor	0%	33.5%	66.5%	0%
Medios	Mejor	0%	44.1%	55.7%	0.2%
	Promedio	0%	27.2%	72.6%	0.2%
	Peor	0%	27.2%	72.4%	0.4%
Altos	Mejor	3%	33.7%	60.6%	2.8%
	Promedio	1.9%	23%	71.9%	3.3%
	Peor	1.5%	23%	71.5%	4.1%

Tabla 6: Porcentajes de distancia para retornos bajos, medios y altos

La Tabla 6 permite observar que:

- Para los casos con retornos bajos, el AG nunca superó a GLPK, pero siempre se mantuvo por debajo del 5% aceptable.
- Para los casos con retornos medios, el AG tampoco superó a GLPK, y solo un 0.2% de las mejores soluciones superaron el 5% aceptable.
- Para los casos con retornos altos, y considerando las mejores soluciones, el AG logró mejorar a GLPK en un 3%, aunque superó el 5% aceptable en un 2.8%.
- A medida que aumenta la cantidad de retornos con respecto a la demanda, las soluciones obtenidas con el AG comienzan a distar más de las soluciones obtenidas con GLPK.
- El 3% de los casos con retornos altos donde el AG superó a GLPK, fueron aquellos donde GLPK no logró encontrar una solución óptima dentro del tiempo permitido.

La Figura 18 muestra las gráficas para la Tabla 6.

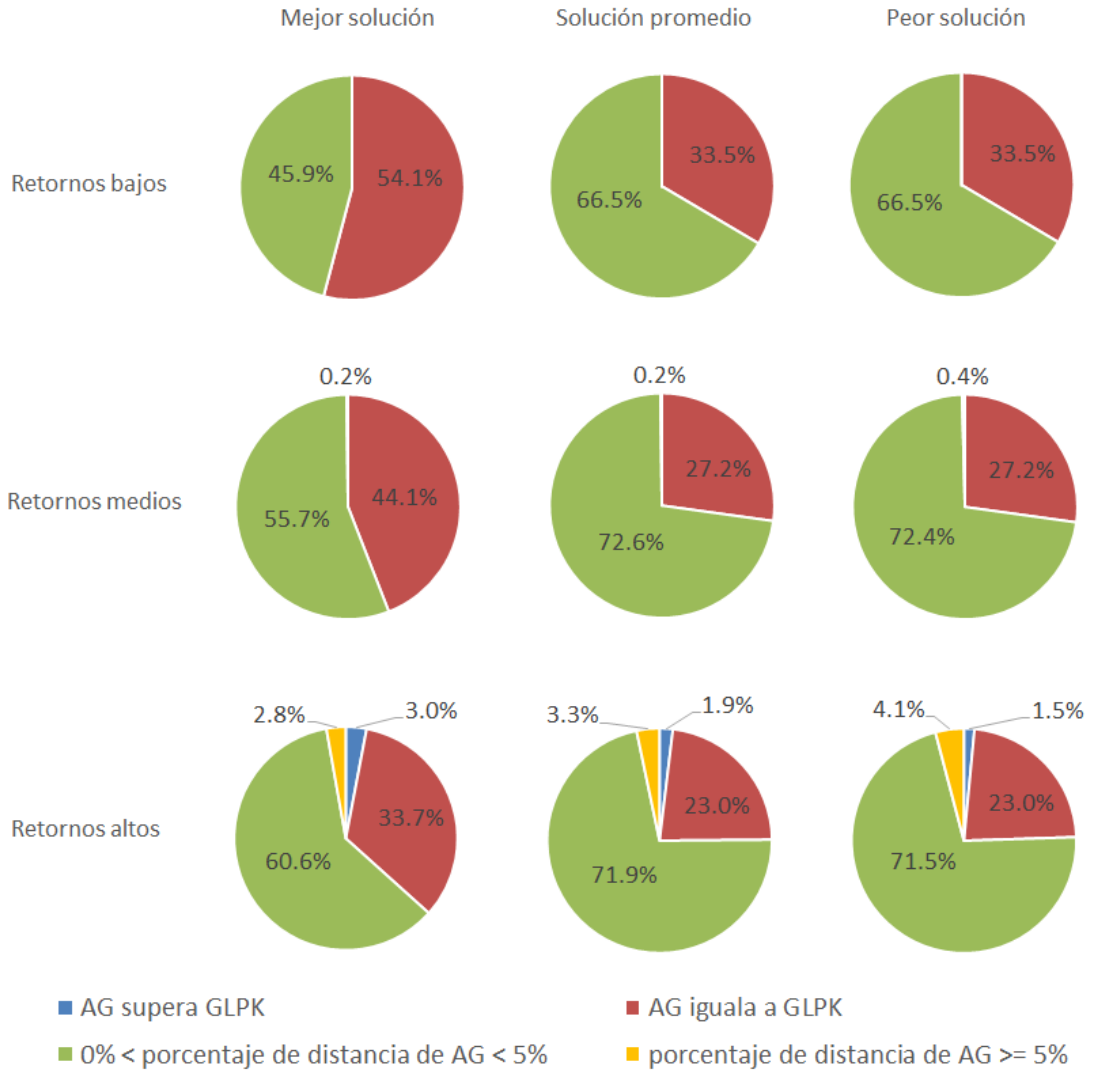


Figura 18: Comparación de AG con GLPK para retornos bajos, medios y altos.

La Figura 19 muestra un ejemplo de la evolución del fitness, a través de las generaciones, para el archivo de entrada *schulz_106_7.dat*:

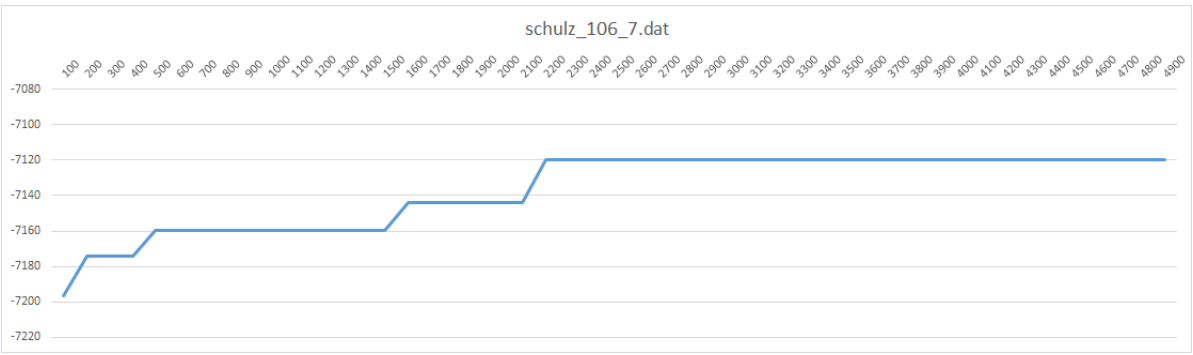


Figura 19: Evolución del fitness para el archivo “schulz_106_7.dat”

La Figura 20 es otro ejemplo de la evolución del fitness para el archivo de entrada "*schulz_3_7.dat*".

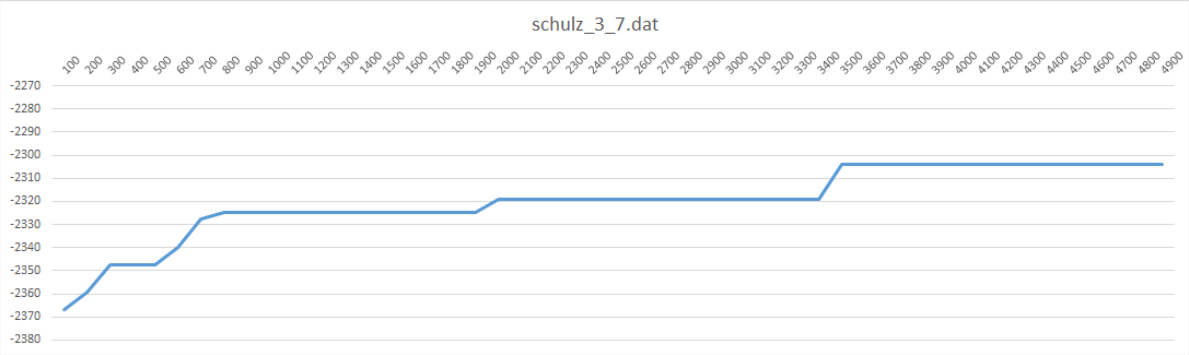


Figura 20: Evolución del fitness para el archivo "schulz_3_7.dat"

En ambos casos se pueden ver los saltos que corresponden a mejoras en el resultado obtenido mientras se realiza la ejecución.

5.3.3. Instancias basadas en Yang et al. (2005)

Los datos basados en Yang et al. (2005) se dividieron en 6 subconjuntos de períodos (10, 12, 14, 16, 18 y 20), como se mencionó anteriormente, y para cada uno de ellos, se generaron 5 archivos de datos.

Recordar que las soluciones para estos casos se consideraron aceptables si el porcentaje de distancia no superaba el 10%.

Dado que en ninguno de los casos (considerando la mejor, promedio y peor solución) el AG superó la solución obtenida en GLPK, pero tampoco estuvo a más del 10% de GLPK, se hizo una distinción intermedia del 5% para tener más información del comportamiento del algoritmo genético.

En el caso del AG el tiempo de cada ejecución fue restringido por la cantidad de generaciones (8000) y el tamaño de la población (500 individuos).

La Tabla 7 compara los tiempos de ejecución entre GLPK y el algoritmo genético. Se observa que, a medida que la cantidad de períodos crece, el AG toma menos tiempo que GLPK, encontrando soluciones aceptables en todos los casos.

Períodos	Tiempo promedio GLPK en segundos	Tiempo promedio AG en segundos
10	0.9	31
12	40	36
14	105	42
16	1740	47
18	1800 (*)	52
20	1800 (*)	56

Tabla 7: Comparación de tiempos entre GLPK y el AG.
(*) Alcanzando el límite máximo de ejecución permitido (30 minutos).

Para los diferentes juegos de datos, GLPK tuvo el siguiente comportamiento:

- Siempre alcanzó la solución óptima en los casos de 10, 12 y 14 períodos.
- Solo alcanzó la solución óptima en el 20% de los casos de 16 períodos.
- Nunca alcanzó la solución óptima para los casos de 18 y 20 períodos.

La Tabla 8 muestra distinguiendo por períodos, los porcentajes donde AG se mantuvo por debajo del 5% de distancia respecto de GLPK, y los porcentajes donde superó el 5% pero se mantuvo por debajo de lo aceptable.

Períodos	Solución	0% < porcentaje de distancia de AG < 5%	5% <= porcentaje de distancia de AG < 10%
10	Mejor	100%	0%
	Promedio	80%	20%
	Peor	40%	60%
12	Mejor	100%	0%
	Promedio	40%	60%
	Peor	20%	80%
14	Mejor	40%	60%
	Promedio	0%	100%
	Peor	0%	100%
16	Mejor	60%	40%
	Promedio	20%	80%
	Peor	20%	80%
18	Mejor	80%	20%
	Promedio	0%	100%
	Peor	0%	100%

20	Mejor	100%	0%
	Promedio	60%	40%
	Peor	0%	100%

Tabla 8: Porcentajes de distancias por períodos.

Para estos casos no se identifica un patrón donde los resultados obtenidos por el AG empeoren o mejoren respecto a los obtenidos en GLPK. Como se observa en la Tabla 8, tanto para los juegos de datos de 10 como para los de 20 períodos (ambos extremos), el total de sus soluciones estuvo a menos del 5% de distancia de GLPK, es decir que la cantidad de períodos no fue un factor que empeorara las soluciones del AG en estos casos.

La Figura 21 muestra las gráficas correspondientes a los valores de la Tabla 8.

Para el total de los juegos de datos, sin distinguir por período y considerando las mejores soluciones, se obtiene que el 80% de los mismos estuvo a menos del 5% de la solución hallada por GLPK, mientras que el restante 20% estuvo entre un 5% y 10%.

Figura 21: Comparación de AG con GLPK para los distintos períodos.

La gráfica de la Figura 22 compara los valores para la mejor solución obtenida por el AG contra los valores de GLPK, para los 30 juegos de datos generados.

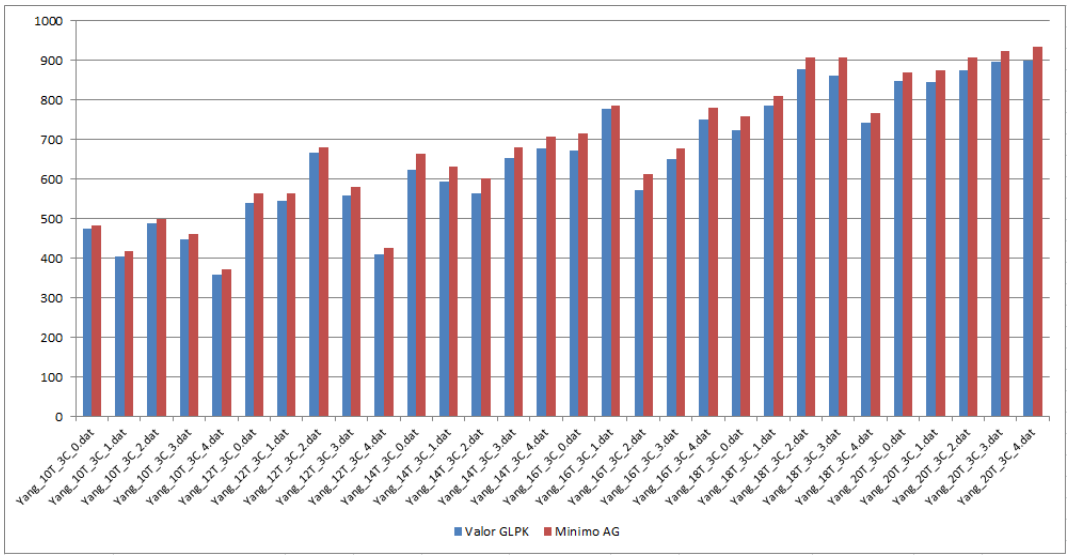


Figura 22: Comparación entre resultados de GLPK y AG.

Como se puede observar, en todos los casos el valor obtenido por el AG es mayor al obtenido por GLPK.

La gráfica de la Figura 23 presenta el porcentaje de distancia para todos los casos. Donde se observa que todos los resultados están por debajo del 10% aceptable. Y solo para 6 de estos casos el porcentaje supera el 5%.

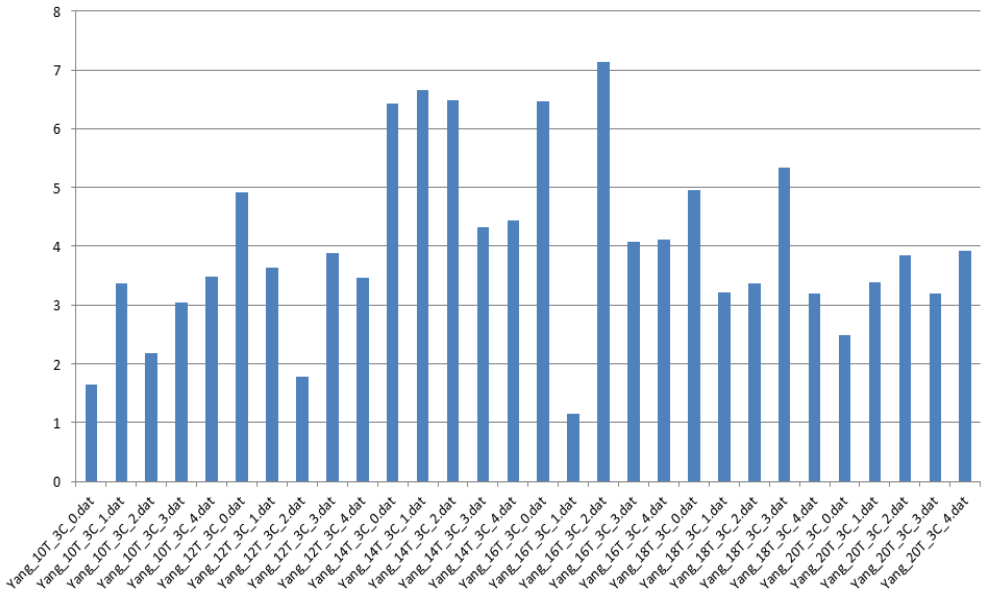


Figura 23: Porcentaje de distancia.

Al igual que en los casos anteriores, también se registró la evolución de fitness para los juegos de datos basados en Yang.

La Figura 24 muestra la evolución del fitness, a través de las generaciones, para el archivo de entrada "*Yang_10T_3C_2.dat*".

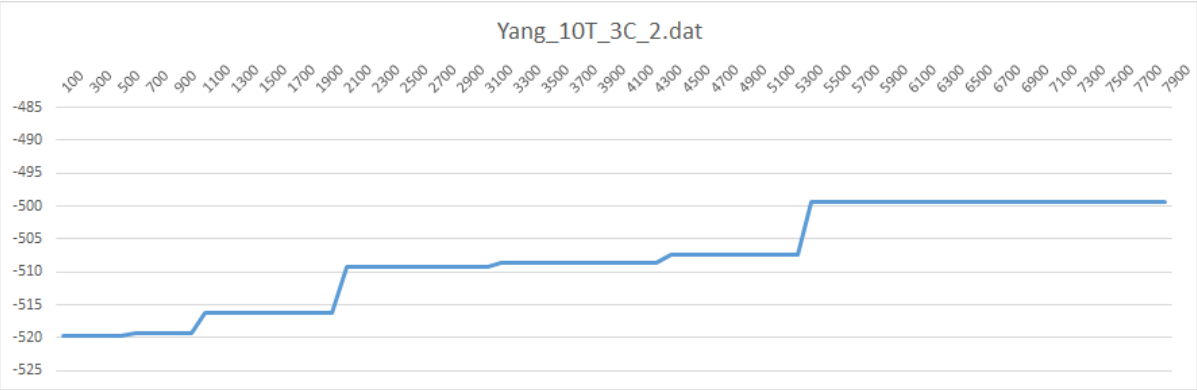


Figura 24: Evolución del fitness para el archivo "Yang_10T_3C_2.dat"

La Figura 25 es otro ejemplo de la evolución del fitness para el archivo de entrada "*Yang_18T_3C_1.dat*".

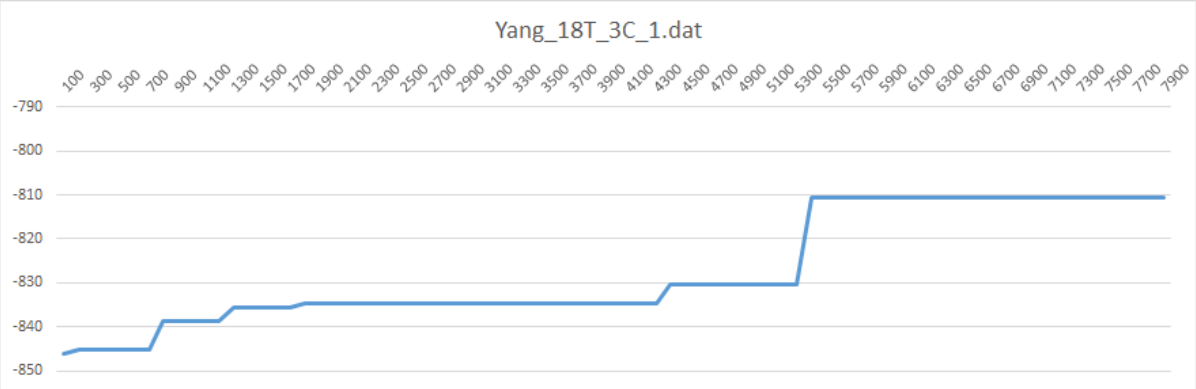


Figura 25: Evolución del fitness para el archivo "Yang_18T_3C_1.dat"

5.3.4. Instancias propias (GLM)

Como se mencionó anteriormente se armaron juegos de datos propios con algunas de las características presentadas en el trabajo de Sifaleras et al. (2015). Los mismos presentan las siguientes características:

- 12 períodos y 3 calidades para la clasificación de los retornos. Se diferenció entre casos de cantidad de retornos bajos, medios y altos, con respecto a la demanda.
- 24 períodos y 4 calidades para la clasificación de los retornos.
- 52 períodos y 3 calidades para la clasificación de los retornos.
- Para los casos de 24 y 52 períodos sólo se consideró una cantidad media de retornos con respecto a la demanda.
- Todos los casos manejan disposición final y venta de retornos.

- Las soluciones para estos casos se consideraron aceptables si el porcentaje de distancia con la solución de GLPK no superaba el 10%.
- Por cada caso se generaron 10 archivos de datos, teniendo un total de 50 casos.

GLPK siempre alcanzó la solución óptima para los juegos de datos de 12 períodos, a diferencia de los casos de 24 y 52. En estos últimos GLPK nunca alcanzó la solución óptima, consumiendo el tiempo máximo de ejecución predefinido, 30 minutos.

En el caso del AG el tiempo de cada ejecución fue restringido por la cantidad de generaciones (5000 para retornos bajos y 8000 para retornos medios y altos) y el tamaño de la población (500 individuos).

La Tabla 9 compara los tiempos de ejecución (en segundos) entre GLPK y el algoritmo genético. Se puede observar que, a mayor complejidad del problema y cantidad de períodos, el tiempo promedio de ejecución del AG aumenta.

Períodos	Retornos	Tiempo promedio GLPK en segundos	Tiempo promedio AG en segundos
12	Bajos	0.02	16.25
	Medios	0.26	37.39
	Altos	6.11	26.10
24	Medios	1800 (*)	86.48
52	Medios	1800 (*)	146.01

Tabla 9: Comparación de tiempos entre GLPK y el AG.
(*) Alcanzando el límite máximo de ejecución permitido (30 minutos).

La Tabla 10 compara los porcentajes de distancia obtenidos con el AG para los casos con 12, 24 y 52 períodos, sólo teniendo en cuenta la cantidad de retornos medios.

Períodos	Solución	0% < porcentaje de distancia de AG < 5%	5% <= porcentaje de distancia de AG < 10%	porcentaje de distancia de AG >= 10%
12	Mejor	100%	0%	0%
	Promedio	90%	10%	0%
	Peor	40%	60%	0%
24	Mejor	0%	100%	0%
	Promedio	0%	80%	20%
	Peor	0%	40%	60%
52	Mejor	0%	20%	80%

	Promedio	0%	20%	80%
	Peor	0%	20%	80%

Tabla 10: Porcentaje de distancia de AG con 12, 24 y 52 períodos, retornos medios.

Es importante recordar que los casos de 12 y 52 períodos manejan 3 calidades, mientras que los casos de 24 períodos manejan 4 calidades.

Se puede observar que, a mayor cantidad de períodos y complejidad de problema, el porcentaje de distancia del AG con respecto a GLPK es mayor. Si bien en los casos de 52 períodos el tiempo de procesamiento del AG fue mucho menor al tiempo de GLPK, las soluciones obtenidas con el AG están alejadas del porcentaje de distancia aceptable.

La Figura 26 muestra las gráficas correspondientes a los valores de la Tabla 10:

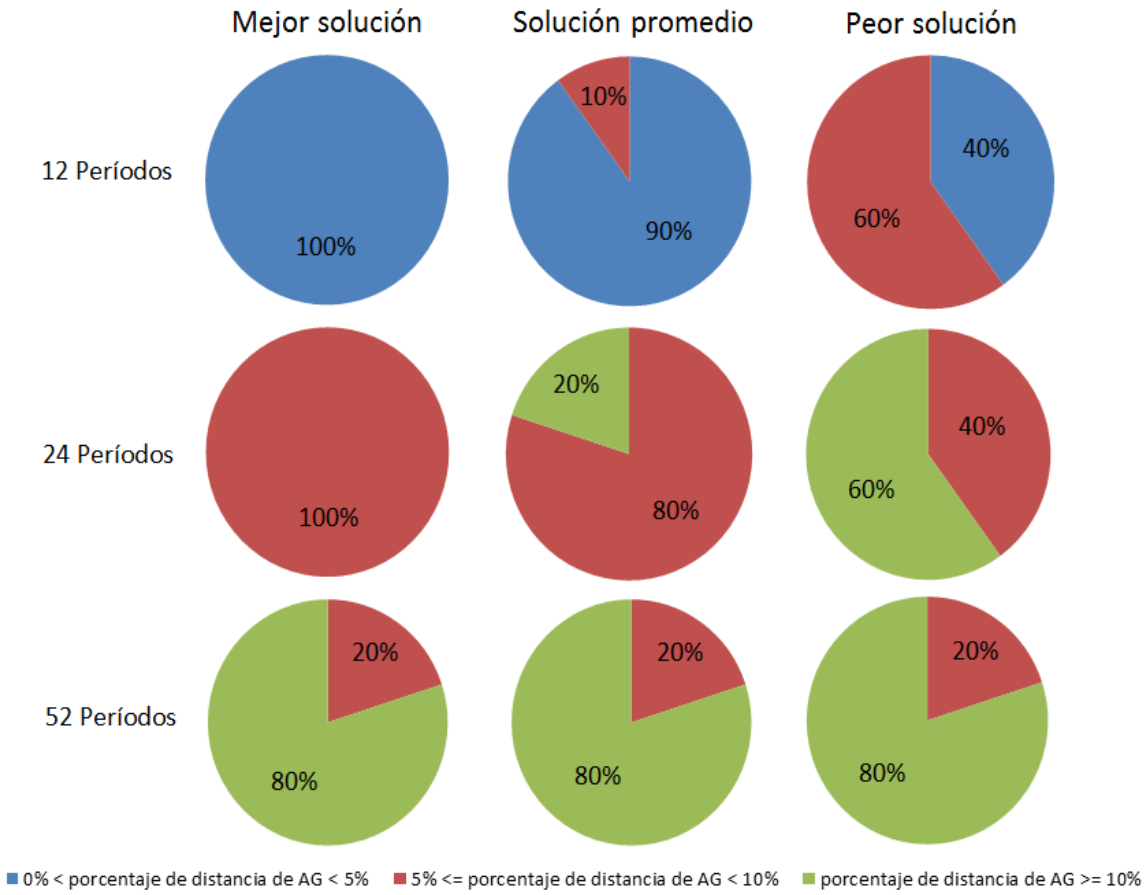


Figura 26: Comparación de AG con GLPK para los distintos períodos.

Para el total de los juegos de datos, sin distinguir por período y considerando las mejores soluciones, se obtiene que:

- el 33.3% de los juegos de datos estuvo a menos del 5% de la solución hallada por GLPK
- el 40% estuvo entre un 5% y 10%
- el 26.7% superó el 10% definido como aceptable

La gráfica de la Figura 27 compara los valores para la mejor solución obtenida por el AG contra los valores de GLPK, para los casos con 12, 24 y 52 períodos, sólo teniendo en cuenta la cantidad de retornos medios. Se puede observar que en todos los casos el valor obtenido por el AG es mayor al obtenido por GLPK.

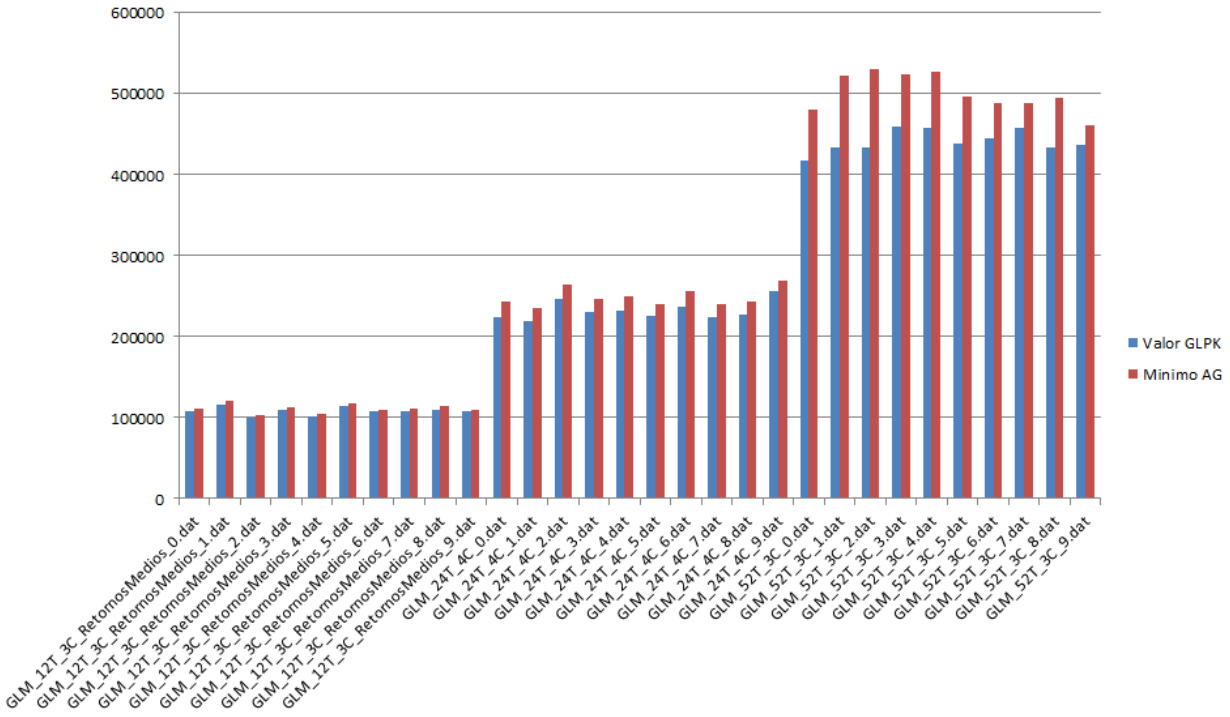


Figura 27: Comparación entre resultados de GLPK y AG.

La gráfica de la Figura 28 presenta el porcentaje de distancia para los juegos de datos anteriores. Como se puede observar, en los casos de 12 y 24 períodos, el porcentaje de distancia estuvo siempre por debajo del valor aceptable, mientras que solo en 2 juegos de datos de 52 períodos se obtiene un valor aceptable.

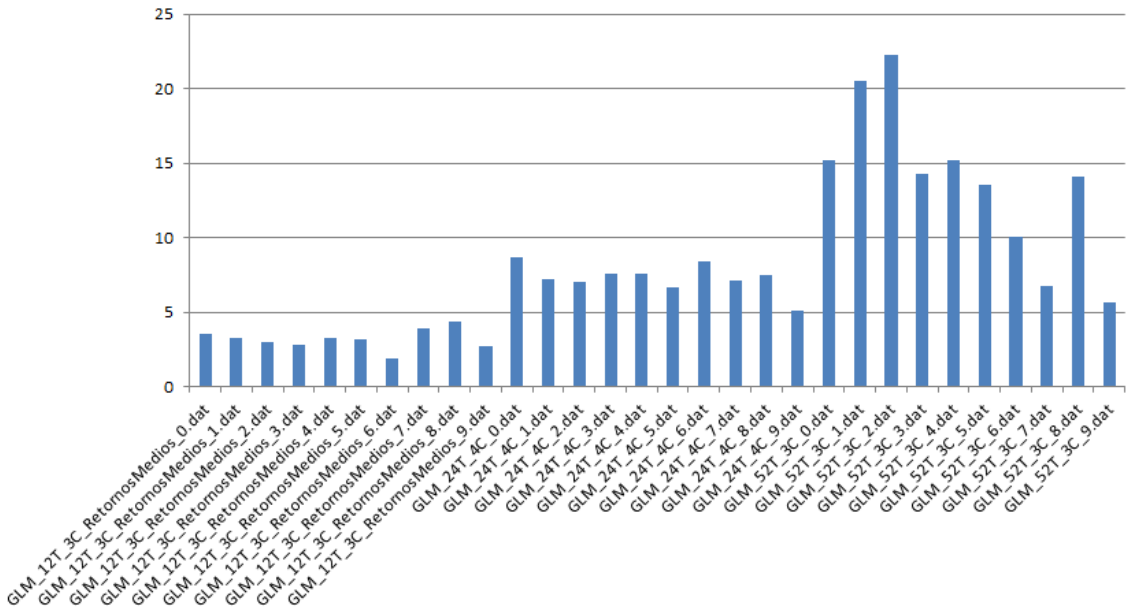


Figura 28: Porcentaje de distancia.

La Tabla 11 compara los porcentajes de distancia obtenidos con el AG para los casos de 12 períodos, diferenciando entre cantidad de retornos bajos, medios y altos con respecto a la demanda.

Retornos	Solución	0% < porcentaje de distancia de AG < 5%	5% <= porcentaje de distancia de AG < 10%	porcentaje de distancia de AG >= 10%
Bajos	Mejor	100%	0%	0%
	Promedio	100%	0%	0%
	Peor	100%	0%	0%
Medios	Mejor	100%	0%	0%
	Promedio	90%	10%	0%
	Peor	40%	60%	0%
Altos	Mejor	50%	50%	0%
	Promedio	10%	90%	0%
	Peor	0%	90%	10%

Tabla 11: Porcentaje de distancia de AG para 12 períodos.

Se puede observar que sólo en el caso de cantidad de retornos altos y considerando la peor solución, el 10% superó el porcentaje de distancia establecido como aceptable (10%). Para el resto de las soluciones (mejor y promedio), si bien el AG no mejoró la solución obtenida por GLPK, los resultados obtenidos siempre estuvieron por debajo del porcentaje de distancia establecido.

La Figura 29 muestra las gráficas correspondientes a los valores de la Tabla 11.

Para el total de los juegos de datos, sin distinguir por cantidad de retornos y considerando las mejores soluciones, se obtiene que:

- el 83.3% de los juegos de datos estuvo a menos del 5% de la solución hallada por GLPK
- el 16.7% estuvo entre un 5% y 10% definido como aceptable

La gráfica de la Figura 30 compara los valores para el mínimo obtenido por el AG contra los valores de GLPK, para los casos para 12 períodos, diferenciando entre cantidad de retornos bajos, medios y altos con respecto a la demanda. Se puede observar que en todos los casos el valor obtenido por el AG es mayor al obtenido por GLPK.

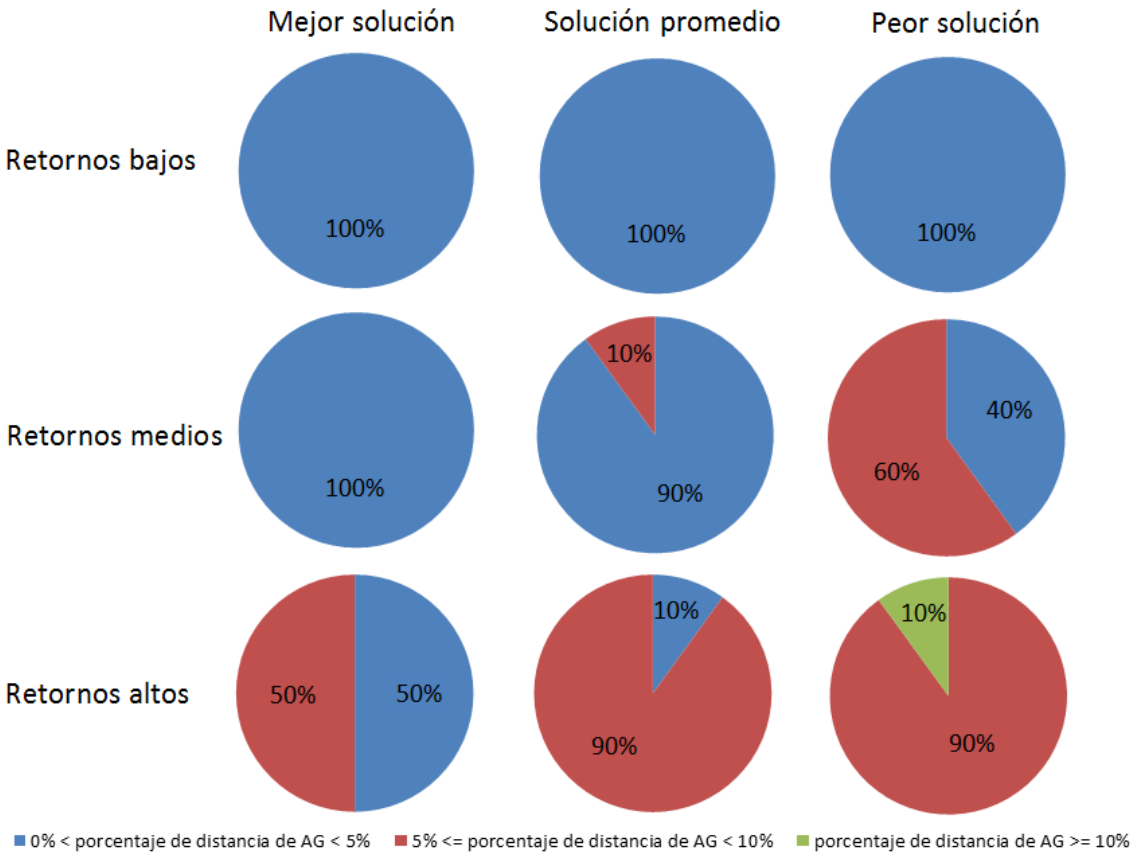


Figura 29: Comparación de AG con GLPK para retornos bajos, medios y altos.

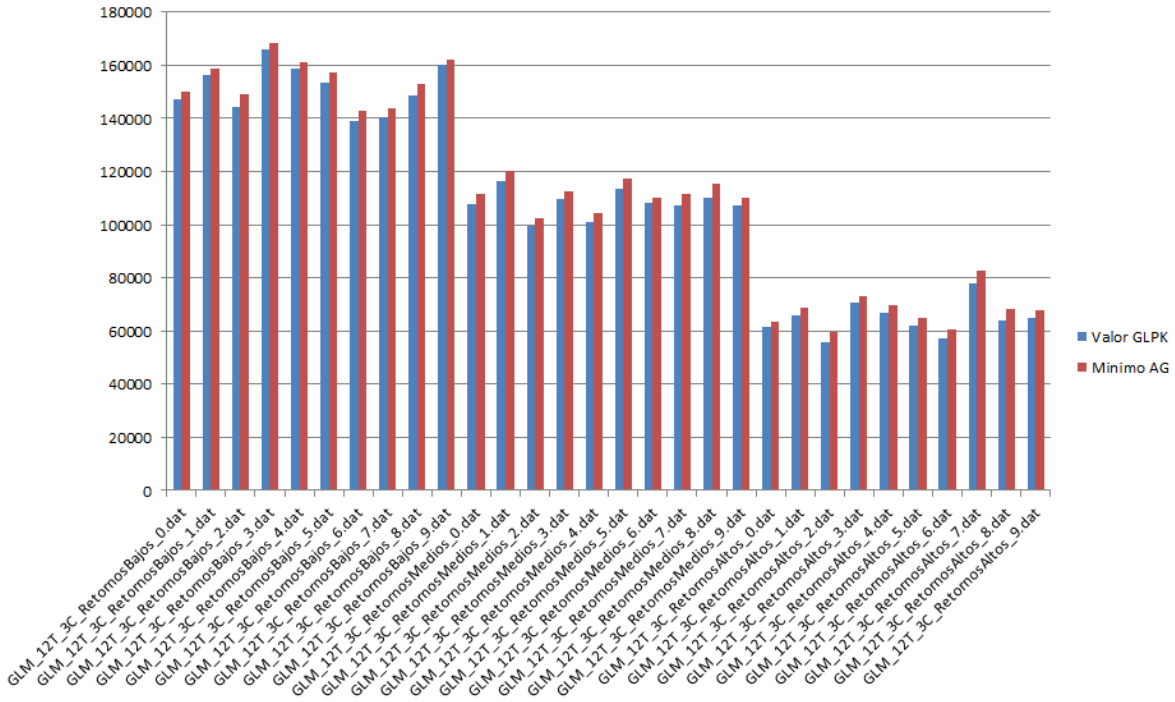


Figura 30: Comparación entre resultados de GLPK y AG.

La gráfica de la Figura 31 presenta el porcentaje de distancia para los juegos de datos anteriores. Como se puede observar, en todos los casos el porcentaje de distancia estuvo por debajo del valor aceptable, registrando en la mayoría de los casos un valor por debajo de 5%.

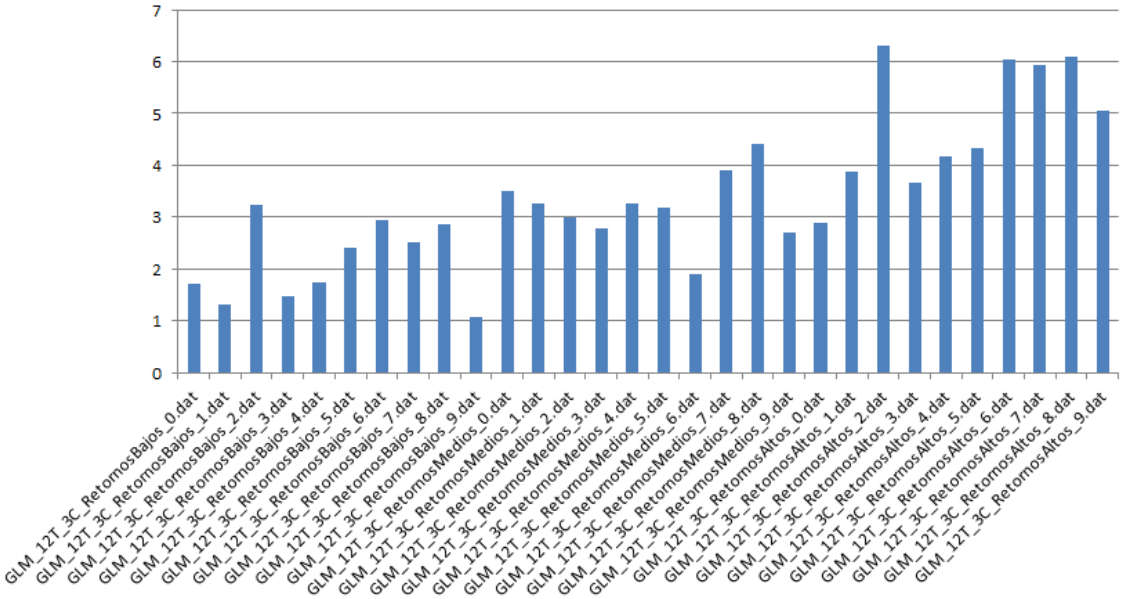


Figura 31: Porcentaje de distancia.

Al igual que en los casos anteriores, también se registró la evolución de fitness para los juegos de datos basados en GLM. Las siguientes figuras muestran la evolución del fitness para los archivos de entrada:

- Figura 32 archivo “GLM_12T_3C_RetornosAltos_6.dat”
- Figura 33 archivo “GLM_24T_4C_4.dat”
- Figura 34 archivo “GLM_52T_3C_2.dat”

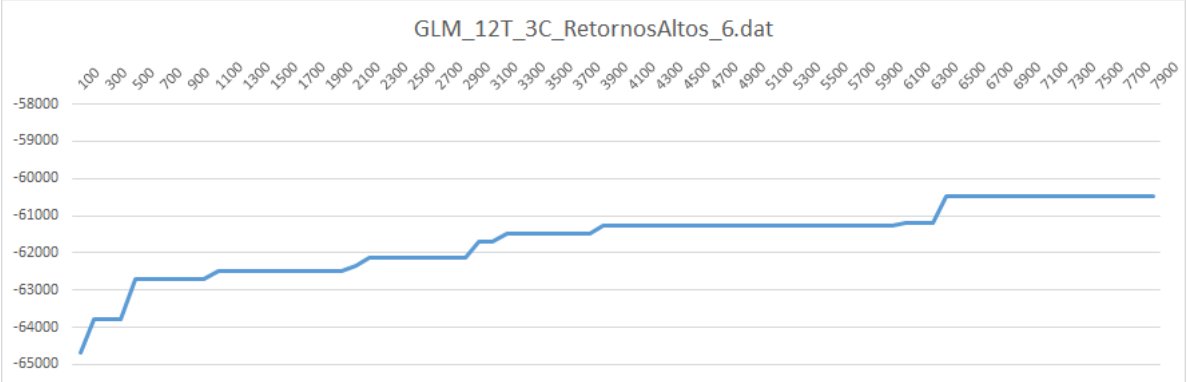


Figura 32: Evolución del fitness para el archivo “GLM_12T_3C_RetornosAltos_6.dat”

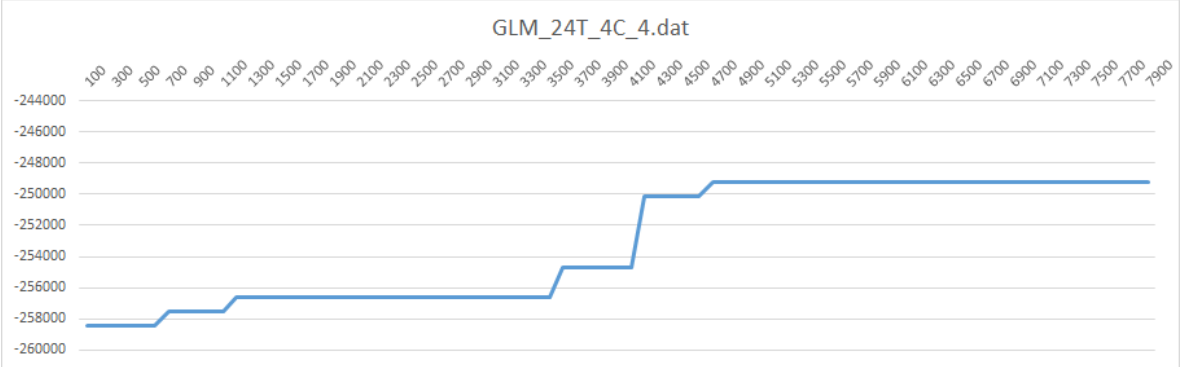


Figura 33: Evolución del fitness para el archivo “GLM_24T_4C_4.dat”

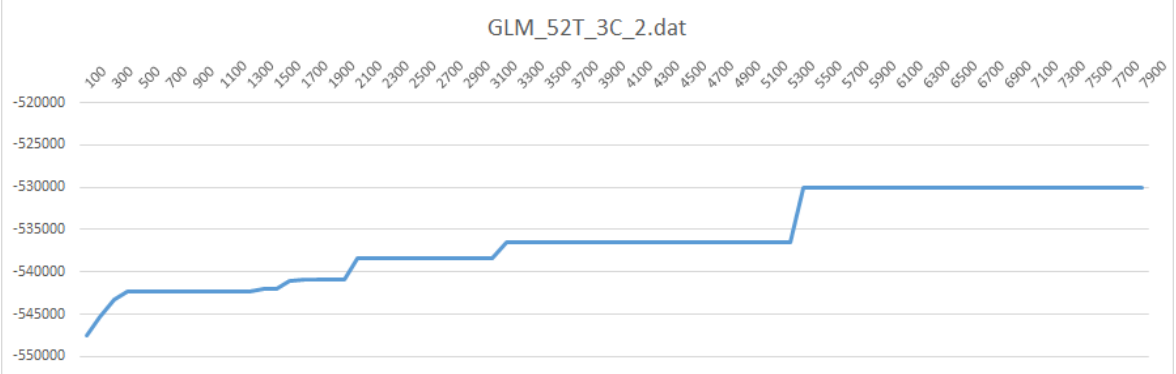


Figura 34: Evolución del fitness para el archivo “GLM_52T_3C_2.dat”

5.3.5. Resumen de análisis de resultados

De las pruebas realizadas, se observó que los mejores resultados obtenidos fueron para aquellas instancias basadas en Sifaleras et al. (2015) y Schulz (2011), donde los costos de configuración son constantes, los costos unitarios son cero, y no hay venta ni disposición final.

De los juegos de datos basados en Sifaleras et al. (2015), en un 82% aproximadamente, la solución obtenida por el AG superó a la obtenida en GLPK, mientras que el restante 18% quedó por debajo del porcentaje de distancia aceptable (5%).

De los juegos de datos basados en Schulz (2011), en un 45% aproximadamente, la solución obtenida por el AG superó o igualó a la obtenida en GLPK, en un 54% estuvo por debajo del porcentaje de distancia aceptable (5%), mientras que solo en un 1% de los casos restantes lo superó.

Distinguiendo los juegos de datos por cantidad de retornos bajos, medios y altos con respecto a la demanda, se observó que las soluciones obtenidas por el AG para retornos bajos nunca superaron el 5% aceptable, mientras que, para retornos medios y altos, solo el 0.2% y 2.8% respectivamente, superaron lo aceptable. Es decir que a medida que la cantidad de retornos aumenta respecto de la demanda, las soluciones obtenidas por el AG comienzan a distar más de las obtenidas por GLPK. Esto puede deberse a la asignación de cantidad de actividades a realizar en la población inicial, donde a mayor cantidad de retornos por períodos, es de esperarse que se realicen más actividades de remanufacturación que de producción con respecto a los casos de retornos bajos. Actualmente no existe en la creación de la población inicial, una distinción por cantidad de retornos respecto a la demanda.

Para los casos basados en Yang et al. (2005), donde los costos unitarios y de configuración son variables, manejan venta de retornos y no consideran disposición final, las soluciones obtenidas por el AG nunca mejoraron las obtenidas por GLPK, sin embargo, se mantuvieron por debajo del porcentaje de distancia que se manejó como aceptable (10%), donde incluso un 80% se mantuvo por debajo del 5% de distancia.

Para los juegos de datos propios (GLM), donde los costos unitarios y de configuración son variables, se manejan venta de retornos y consideran disposición final, las soluciones obtenidas por el AG nunca mejoraron las obtenidas por GLPK. Para los casos de 52 períodos, solo el 20% se mantuvo por debajo del porcentaje aceptable (10%). Para los casos de menos períodos, las soluciones obtenidas por el AG siempre se mantuvieron por debajo del límite aceptable.

Distinguiendo los juegos de datos por retornos bajos, medios y altos, se observó que las soluciones obtenidas por el AG para retornos bajos y medios siempre se mantuvieron por debajo del 5% de distancia, mientras

que, para retornos altos, todas las soluciones quedaron por debajo del 10% aceptable, donde la mitad de éstos no superó el 5% de distancia. En estos casos puede observarse el mismo comportamiento que para los casos basados en Schulz (2011), donde a mayor cantidad de retornos con respecto a la demanda, las soluciones del AG distan más de las soluciones de GLPK.

En cuanto a los tiempos de ejecución, el AG siempre estuvo restringido por la cantidad de generaciones y el tamaño de la población, tomando tiempos similares en las distintas ejecuciones para una misma configuración. GLPK, en cambio, se restringió a 30 minutos de ejecución si no encontraba una solución óptima antes, haciendo que variara su tiempo de ejecución según la complejidad del problema.

Para el AG, si bien los juegos de datos con mayor cantidad de períodos registraron menos tiempo de ejecución en comparación con GLPK, los resultados obtenidos no siempre fueron mejores. En la mayoría de los casos basados en Sifaleras et al. (2015) se logró superar las soluciones halladas por GLPK, contrariamente a los casos propios de la misma cantidad de períodos (52). Nótese que la complejidad de los juegos de datos propios es mayor ya que manejan más parámetros.

6. Conclusiones

En este proyecto se estudió el problema de control de inventario con clasificación, remanufacturación, venta y disposición de retornos, extendiendo el ya conocido problema ELSR, al cual se denomina ELSRC.

Se cubrió el alcance propuesto para el proyecto, que abarcó las siguientes actividades: i) estudio de parte de la literatura existente que aborda diferentes problemas de inventario, haciendo foco en los problemas con opciones y clasificación de retornos, ii) creación del documento del estado del arte, donde se describen y comentan los trabajos consultados, iii) creación del modelo matemático para el problema, iv) implementación de dicho modelo en GLPK, v) desarrollo y evaluación de un procedimiento basado en alguna metaheurística para solucionar el problema.

La diferencia principal entre este proyecto y los trabajos consultados de la literatura, es que se tuvieron diferentes características del problema en un mismo modelo: producción y remanufacturación de retornos para cubrir la demanda, clasificación de los retornos en diferentes inventarios según su nivel de calidad, eventual venta y disposición de retornos que no son remanufacturados.

Se mostró que el ELSRC es un problema NP-hard, ya que se puede considerar una extensión del ELSR. Teniendo esto en mente, se diseñó, implementó y evaluó un procedimiento de resolución basado en la metaheurística de Algoritmos Genéticos (AG). Además, se implementó el modelo en GLPK para obtener soluciones contra las cuales comparar el procedimiento AG.

Durante la implementación del AG se tuvieron algunas dificultades para obtener soluciones de buena calidad.

Para juegos de datos con costos unitarios fijos o variables, el AG no llegaba a una buena solución en todos los casos. Esto se debía a cómo se calculaba la cantidad de retornos a remanufacturar en cada período. Analizando los resultados obtenidos en GLPK se observó que, para los casos de costos unitarios fijos, las soluciones no realizaban más de una actividad (remanufacturación o producción) por período, y en caso de remanufacturar solo lo hacían para una calidad. Para problemas de costos unitarios variables, existían períodos donde se realizaba más de una actividad, y en caso de remanufacturar, podía ocurrir para retornos de distintos niveles de calidad. Se decidió entonces dividir el comportamiento del algoritmo en dos: problemas con costos unitarios fijos y problemas con costos unitarios variables.

De las observaciones sobre los resultados obtenidos por GLPK, se aplicaron otras técnicas para mejorar el AG. Al generar la población inicial se

tuvo en cuenta la frecuencia de las actividades, de modo de comenzar con un espacio de soluciones más cercano a la solución de GLPK. Se realizó un ajuste para agrupar actividades de producción, reduciendo los costos de configuración, y otro en el inventario de productos listos para que en el último período su nivel sea cero.

Al momento de remanufacturar, se varió la política de selección de retornos para los diferentes niveles de calidad, intercambiando entre: elegir retornos de mejor a peor calidad y viceversa, o elegir retornos de forma aleatoria.

Se parametrizó la posibilidad de manejar venta y/o disposición final de retornos, permitiendo reducir los tiempos de procesamiento del AG.

Los juegos de datos utilizados para el estudio de resultados fueron basados en los trabajos de Sifaleras et al. (2015), Schulz (2011) y Yang et al. (2005), debido a que se contaba con información de las pruebas realizadas en dichos trabajos y los mismos tenían diferentes niveles de complejidad. Además, se crearon juegos de datos propios, combinando características de los trabajos anteriores y agregando nuevas, como diferentes niveles de calidad en los retornos. Los mismos tuvieron casos de 12, 24 y 52 períodos, 3 y 4 niveles de calidad, cantidad de retornos altos, medios y bajos con respecto a la demanda, con venta de retornos y disposición final, y costos variables.

Luego de hacer pruebas con un conjunto reducido de datos, se eligió la mejor configuración paramétrica para cada juego de datos.

En total se crearon 1.916 juegos de datos que se ejecutaron en GLPK y el AG implementado.

Se consideró una solución aceptable si el algoritmo genético alcanzaba una solución que distara menos del 5% de la solución hallada por GLPK para los casos de costos unitarios fijos, o 10% para los casos de costos unitarios variables.

Luego de analizar los resultados, se observó que: para costos unitarios fijos el AG mejora a GLPK en el 18.5% de los casos, y devuelve un resultado aceptable en el 80.6%; mientras que para costos unitarios variables el AG no mejora a GLPK, pero devuelve un resultado aceptable en el 90% de los casos. El 10% restante, corresponde a juegos de datos con gran cantidad de períodos (52). Esto puede ser un buen punto de partida para trabajos futuros.

7. Trabajos futuros

A continuación, se describen algunas posibles mejoras y nuevas funcionalidades para la solución implementada:

- Estudiar en profundidad el comportamiento para los casos con costos unitarios variables y muchos períodos con la intención de mejorar los resultados obtenidos por el AG para estos casos.

- Generar mayor cantidad de juegos de datos basados en Yang et al. (2005) para poder identificar un patrón en el comportamiento obtenido.

- Mejoras técnicas en el algoritmo:
  - Quitar la venta de la representación del individuo cuando no se maneja la misma. De esta forma, un individuo tendría menos alelos y en consecuencia habría menos tiempo de procesamiento en operaciones como mutación o cruzamiento.
  - Quitar la verificación de factibilidad al calcular el fitness ya que dicha verificación queda obsoleta por cómo se calculan las cantidades por actividad de los individuos, reduciendo además tiempos de procesamiento.
  - Parametrizar frecuencias de actividades en la generación de la población inicial y además poder diferenciar las mismas según cantidad de retornos bajos, medios y altos con respecto a la demanda.

- Extender el Analizador de Resultados desarrollado, para profundizar en cálculos de probabilidad y estadística a través de su librería incorporada R.NET versión 1.5.13 [6]. De esta manera, sería posible aplicar test estadísticos tales como test de normalidad e hipótesis.

8. Índice de anexos

9. Referencias

[1] Algoritmos Evolutivos, 2016. *Instituto de Computación, Facultad de Ingeniería*. https://www.fing.edu.uy/inco/cursos/ae/pmwiki/pmwiki.php?n=Main.Material, 22/11/2016

[2] Algoritmos Genéticos
http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf, 22/11/2016

[3] Álvarez JP, 2010. La remanufactura, una nueva oportunidad para la industria y el comercio exterior en México, pp. 119.
http://www.dofiscal.net/pdf/doctrina/D_DPF_RV_2010_185-A17.pdf, 22/11/2016

[4] Biblioteca ECJ,
https://cs.gmu.edu/~eclab/projects/ecj/, 22/11/2016

[5] Biblioteca Mallba,
http://neo.lcc.uma.es/mallba/easy-mallba/index.html, 22/11/2016

[6] Biblioteca R.NET,
https://rdotnet.codeplex.com/, 22/11/2016

[7] BMW Group,
https://www.bmwgroup.com/content/dam/bmw-group-websites/bmwgroup_com/responsibility/downloads/en/2015/BMW_SVR_2015_RZ_EN.pdf, 22/11/2016

[8] Caterpillar,
http://www.caterpillar.com/, 22/11/2016

[9] Denizel M, Ferguson M, Souza G, 2007. Multi-Period Remanufacturing Planning With Uncertain Quality of Inputs. *Engineering Management*, 57(3), pp. 394 – 404.

[10] Ferguson M, Guide D, Koca E, Souza G, 2006. Remanufacturing Planning with Different Quality Levels for Product returns. *Robert H. Smith School Research* Paper No. RHS 06-050.

[11] Ferrer E, Mouriz A, Piñeyro P, 2002. Políticas de inventarios, mantenimiento y pronósticos. Anexo B Control de Inventario. Taller V, *Departamento de Investigación Operativa, Instituto de Computación, Facultad de Ingeniería*.

[12] Goren HG, Tunali S, Jans R, 2010. A review of applications of genetic algorithms in lot sizing. *Journal of Intelligent Manufacturing*, 21(4), pp. 575-590.

[13] Gutowski T, Sahni S, Boustani A, Graves S, 2011. Remanufacturing and Energy Savings. *Environmental Science & Technology*, 45(1), pp. 4540–4547.

[14] Lenguaje R,
https://www.r-project.org/, 22/11/2016

[15] Logística Inversa,
https://es.wikipedia.org/wiki/Log%C3%ADstica_inversa, 22/11/2016

[16] Mahapatra S, Pal R, Narasimahan R, 2012. Hybrid (re)manufacturing: manufacturing and operational implications. *International Journal of Production Research*, 50(14), pp. 3786-3808.

[17] Manual de ECJ,
https://cs.gmu.edu/~eclab/projects/ecj/docs/manual/manual.pdf , 22/11/2016

[18] Nenes G, Nikolaidis Y, 2012. A Multi–period Model for Managing Used Product Returns. *International Journal of Production Research*, 50(5), pp. 1360-1376.

[19] Piñeyro P, Viera O, 2009. Inventory policies for the economic lot-sizing problem with remanufacturing and final disposal options. *Journal of Industrial and Management Optimization,* 5(2), pp. 217–238

[20] Schulz T, 2011. A new Silver–Meal based heuristic for the single-item dynamic lot sizing problem with returns and remanufacturing. *International Journal of Production Research*, 49(9), pp. 2519-2533.

[21] Sifaleras A, Konstantaras I, Mladenović N, 2015. Variable neighborhood search for the economic lot sizing problem with product returns and recovery. *Int. J. Production Economics*, 160(1), pp. 133–143.

[22] Sitio oficial de GLPK,
https://www.gnu.org/software/glpk/, 22/11/2016

[23] Teunter R, Bayindir Z, Van Den Heuvel W, 2006. Dynamic lot sizing with product returns and remanufacturing. *International Journal of Production Research*, 44(20), pp. 4377-440Glosario

[24] Unilever,
https://www.unilever.com.uy, 22/11/2016

[25] Unilever - Cero relleno sanitario de alimentos,
http://www.anii.org.uy/apoyos/innovacion/desafios/2/cero-relleno-sanitario-de-alimentos--unilever/, 22/11/2016

[26] Xerox,
http://www.xerox.com/corporate-citizenship/2011/sustainability/waste-prevention.html, 22/11/2016

[27] Yang J, Golany B, Yu G, 2005. A Concave-Cost Production Planning Problem with Remanufacturing Options. *Naval Research Logistics*, 52(1), pp. 443–458.

Control de Inventario con Remanufacturación y Clasificación de Retornos

Estado del arte

Proyecto de grado

Luciano Alvarez
Gabriela Arriola
Matilde Macciò

Supervisor: Pedro Piñeyro

Índice

1. Introducción

Debido al crecimiento en el área industrial y comercial, las empresas se enfrentan a la necesidad de mantener y controlar los niveles de inventario de sus productos para reducir costos y maximizar ganancias.

Existen diferentes motivaciones para realizar controles de inventario [10]. Algunos de ellas son: variaciones en las demandas, limitaciones en capacidad de producción, limitaciones de espacio para depósitos, tiempos y costos de poner en producción un tipo de artículo, necesidades de mantener un inventario de anticipación previendo reducir costos, necesidades en algunos casos de mantener un inventario interno para alimentar una cadena de producción. Esto requiere una investigación y posterior planificación para poder evitar grandes pérdidas económicas en las organizaciones.

Es por eso que el control de inventario ayuda a determinar cuánto y cuándo producir, para satisfacer la demanda de uno o varios productos, cumpliendo ciertas restricciones y minimizando la suma de los costos involucrados [10].

En los últimos años varias empresas han comenzado a involucrarse en la remanufacturación de los productos usados (de ahora en más los llamaremos retornos). El objetivo de esto es reducir los costos de operación, reducir el uso de nuevas piezas y de materias primas, minimizando los desperdicios y la contaminación generada. El hecho de que una empresa decida no remanufacturar puede hacer que los retornos se dirijan a otras empresas, donde sí sean remanufacturados, afectando la cuota de mercado y los ingresos del fabricante original [15].

Generalmente la cantidad de productos retornados a remanufacturar no alcanza a cubrir la demanda, por lo cual es importante poder llevar adelante y planificar ambas tareas: manufacturación y remanufacturación.

Los retornos deben reunir ciertas características para que puedan ser remanufacturados [6]. Es necesario que pasen por un proceso de inspección y posterior clasificación ya que el nivel de calidad con el que llegan puede ser muy heterogéneo y el costo de remanufacturar varía como consecuencia. Estos retornos seleccionados para satisfacer la demanda, pasan por un proceso de remanufacturación que los restaura, dejándolos en iguales condiciones de calidad y funcionamiento que el producto original. En caso de que los retornos no sean remanufacturados, pueden ser vendidos o desechados.

2. Objetivo

El foco del estado del arte, en el marco del proyecto de grado es abordar el problema de control de inventario con remanufacturación y clasificación de retornos. Pudiendo determinar las cantidades a producir, remanufacturar, desechar y vender en cada período, cumpliendo los requerimientos de la demanda y minimizando los costos de las actividades a realizar.

Para esto se han consultado diferentes fuentes enfocándonos en los trabajos que tienen características similares al problema que vamos a resolver en éste proyecto. Dado que la literatura disponible es muy amplia, quedan fuera del alcance de este documento otros problemas de inventario no menos importantes, como por ejemplo: más de un artículo, demanda estocástica, revisión de nivel de inventario continuo, entre otros.

Este documento se organiza en las siguientes secciones. En la Sección 3 se definen los principales conceptos, componentes, características del control de inventario, y se definen los conceptos de remanufacturación y clasificación de retornos. En la Sección 4 se presentan diferentes modelos para control de inventario, comenzando desde los modelos más básicos y extendiéndolos a modelos con características más complejas. En la Sección 5 se presentan las conclusiones.

3. Control de inventario

Se define como **inventario** [10] al conjunto de productos (o artículos) que se deben mantener en algún lugar por un período de tiempo determinado y de los cuales se espera obtener algún beneficio.

Tenemos un **problema de inventario** [10] cuando:

- Existen limitaciones físicas: el espacio de depósito, la capacidad de compra o producción es limitada.
- Existen limitaciones de producción: debido a los tiempos y costos de preparar los equipos para la producción de un determinado artículo, en algunos casos, es conveniente producir más de lo necesario.
- Es necesario un inventario de seguridad: para satisfacer la demanda de un cliente en cualquier momento, evitar el desabastecimiento. Esto se da cuando los tiempos de entrega y la demanda son inciertos.
- Se mantienen otros inventarios, por ejemplo inventarios internos, inventarios de anticipación que preveen reducir costos.
- Hay una gran incertidumbre en cuanto a la venta y a la demanda.

Se entiende por **control de inventario** [10] al conjunto de técnicas que ayudan a determinar cuándo y cuánto adquirir de un determinado artículo, para mantener un nivel de inventario. También involucra la optimización del inventario, en la que los costos deben ser minimizados con el objetivo de maximizar el resultado financiero para la empresa. Un control de inventario bien gestionado, es por lo general la clave para alcanzar los objetivos del margen de beneficio.

3.1. Componentes de inventario

A continuación se definen algunos de los principales componentes del modelo de inventario [10] y en particular los vinculados al caso con opciones de retornos.

- Tiempo de Entrega: es el tiempo transcurrido entre una solicitud para reabastecer el inventario y el momento en que llegan los insumos. Dicho tiempo puede ser modelado como determinístico cuando es conocido (caso más utilizado en los modelos), o estocástico cuando se calcula en función de una distribución de probabilidad. Generalmente y para simplificar los problemas, este tiempo se supone cero (entrega instantánea).
- Descuento por cantidad: por lo general, cuanto mayor sea el pedido realizado a un proveedor, menor será el precio unitario de un artículo. Una consecuencia de esto es que aumente el costo de mantener en inventario.
- Costo de producción: es el costo asociado a la fabricación de un artículo.
- Costos de mantener en inventario: es el costo asociado a almacenar en inventario un artículo, por un lapso de tiempo, hasta que el mismo sea vendido, utilizado o desechado.
- Costo por faltante (penalización): es el costo que se produce cuando la demanda no es satisfecha en un período, es decir, la cantidad de

artículos que hay en inventario no es suficiente para satisfacer la demanda. Pueden existir casos en los cuales esto no signifique un problema y se puedan satisfacer los requerimientos con la producción de períodos futuros. En caso de permitir faltantes, se consideran dos casos:
    - Nivel de inventario positivo: el nivel de inventario es siempre mayor o igual a cero. En caso de no cubrir un pedido con el nivel de inventario disponible, se realiza una producción urgente o se da la venta por pérdida.
    - Nivel de inventario negativo: si no se logra cubrir un pedido con el nivel de inventario actual, la demanda no se pierde y queda pendiente para ser satisfecha en un período posterior (backlogging).
- Costo de remanufacturación: es el costo asociado a la remanufacturación de un retorno. Este costo puede variar dependiendo de la calidad del mismo.
- Costo de disposición final: es el costo asociado a la eliminación de un retorno que no será remanufacturado.
- Costos de configuración: es el costo asociado a la preparación de las tareas de producción, remanufacturación o disposición final.
- Valor de rescate: es la ganancia asociada a la venta de un artículo usado que no ha sido remanufacturado.

3.2 Características de los modelos de inventario

Los modelos de inventarios se pueden clasificar de acuerdo a diferentes criterios. A su vez estos criterios se pueden combinar entre sí [10].

Según la cantidad de artículos se pueden clasificar en modelos de inventario de **un artículo** o de **varios artículos**.

Según el comportamiento de la demanda se pueden clasificar en **demanda determinística**: cuando se conoce o se asume conocida la demanda de un artículo en un período, o en **demanda estocástica**: cuando la demanda se comporta como una variable aleatoria con distribución conocida.

En el caso de modelos de inventario para varios artículos, la demanda de los artículos puede ser considerada **independiente**, si no hay relación entre los artículos, o **dependiente**, si la demanda de un artículo afecta la de otro.

Por revisión del nivel de inventario se pueden clasificar en modelos de **revisión continua**: donde el nivel de inventario se puede determinar en cualquier momento, o en modelos de **revisión periódica**: donde el inventario se revisa cada cierto de tiempo o en un momento particular en cada período.

Según el horizonte de planeación, el modelo de inventarios se puede clasificar dentro de un **horizonte finito**, en caso que la cantidad de períodos sea limitada, o infinito si no.

3.3 Inventario con remanufacturación

En las últimas décadas, términos como reciclaje, reutilización, reducción de recursos, responsabilidad ambiental y fabricación de productos verdes han comenzado a ser familiares para todos nosotros. Los problemas ambientales

relacionados con la actividad industrial y el uso incontrolado de recursos naturales ponen en peligro el desarrollo de futuras generaciones. En vista de ésta situación, algunos países han incorporado legislaciones ambientales para la recuperación de los productos o una correcta eliminación de los mismos (cuando no cumplen con ciertos requisitos mínimos para la recuperación).

La gestión de las actividades relacionadas con el flujo de retorno del consumidor al productor se conoce como Logística Inversa. La misma consiste en cómo los productos son recogidos del usuario final y regresan a una instalación del fabricante para su reparación, reacondicionamiento, reciclaje o destrucción (de manera adecuada). El fabricante se encarga de prestar servicios de transporte y ofrecer descuentos para la renovación de productos, con el fin de hacerlo más llamativo a los consumidores [5, 13].

Dentro de las tareas de recuperación de artículos usados, la **remanufacturación** se define como la recuperación de los productos devueltos. La definición de remanufacturación de Ijomah [16] comprende las tareas de: desmontaje, limpieza, pruebas, operaciones de reemplazo y reparación parcial, reensamblaje, restaurando el producto a su calidad original. Para la mayoría de los enfoques estudiados la remanufacturación ofrece beneficios a todas las partes involucradas. Para el fabricante, éste proceso es menos costoso que fabricar un producto de cero (ahorro en consumo de energía, materias primas y mano de obra). Desde el punto de vista del consumidor, el resultado final es un artículo tan bueno como uno nuevo pero más barato ya que los productos remanufacturados se venden entre el 50% y 80% del valor de un nuevo producto, significando un ahorro para el consumidor [14, 23].

La tasa de retorno de un producto es menor que la tasa de demanda, por lo que no es posible satisfacer toda la demanda del mercado a través de la remanufacturación [13].

En cada período una empresa debe decidir qué hacer con los retornos: desecharlos (de una manera adecuada), almacenarlos para próximos períodos, venderlos o remanufacturarlos. Cada una de estas operaciones tiene un costo asociado, pero no todas significan un gasto para la empresa, dado que la venta de un retorno (rescate) significa un ingreso [15]. Es importante destacar que no todos los retornos tendrán la misma calidad.

El hecho de que un fabricante decida no remanufacturar puede significar un problema, dado que puede estar compitiendo contra sus propios productos remanufacturados por otra empresa.

Otro enfoque sobre la remanufacturación presentado en Gutowski et al. (2011) [15] plantea que no siempre remanufacturar es más conveniente. En aquellos casos donde los productos tienen una fase de uso energético, la remanufacturación se puede tornar más costosa que la manufacturación, pues se debe lograr que el producto retornado (y seguramente degradado), alcance las mismas condiciones de eficiencia energética que un producto nuevo. Esto sucede cuando las nuevas versiones tienen cambios importantes en su arquitectura o tecnología.

Para aquellos productos que no tienen fase de uso energético, o que mantienen incambiado los requisitos sobre la eficiencia en el uso de energía, la remanufacturación puede ahorrar costos energéticos.

3.4 Clasificación de retornos

El éxito de la remanufacturación es un desafío importante para una empresa, debido a la dificultad en la predicción de la demanda para productos remanufacturados, la cantidad no uniforme de retornos y su calidad heterogénea. En particular, los dos últimos conducen a un aumento en los costos operativos de la empresa [13].

Los productos retornados deben ser clasificados en diferentes categorías según su calidad. Los costos de la remanufacturación serán variados dependiendo de dicha calidad. Los retornos de mejor calidad tendrán un costo de remanufactura menor a los retornos de peor calidad. Por otro lado, puede resultar más costoso para la empresa, mantener en inventario los retornos de mejor calidad, por lo que deberá utilizarlos primero al momento de remanufacturar [6].

4. Modelos de control de inventario

Existen diferentes modelos para el control de inventario. Estos difieren según las características que tenga el problema: si es de uno o más artículos, si la demanda es determinística o estocástica, si la revisión del inventario es periódica o continua en el tiempo, entre otros.

En esta sección se presentarán distintos modelos de control de inventario referentes a nuestro foco de estudio: para un solo artículo, con demanda determinística y revisión periódica.

4.1. Política de Tamaño de Lote Económico (EOQ)

Esta política fue propuesta por Harris 1913 [11]. Es uno de los modelos clásicos, más antiguos, base del control de inventarios. Esta política supone demanda conocida y uniforme, sin restricciones de capacidad de producción, sin descuentos por cantidad, con factores de costos fijos conocidos, no permite faltantes y toma el tiempo de entrega como nulo. Puede ser aplicado a un solo artículo, o a varios independientes entre sí.

La misma consiste en producir una cantidad $Q > 0$ cada vez que el nivel de inventario sea cero, repitiéndose el procedimiento de forma cíclica en el largo del tiempo sobre el que se desea planificar.

Los componentes del modelo son [19]:

- P: costos unitarios de producción
- Q: cantidad a producir
- Q^*: cantidad óptima a producir
- D: demanda
- K: costo fijo de puesta en marcha la producción
- h: costo unitario de mantenimiento en inventario de un artículo

La función de costos totales se define como:

$$TC = PD + \frac{DK}{Q} + \frac{hQ}{2} \quad (1)$$

Donde el primer término, $P*D$, significa el costo de producir para la demanda dada.

El segundo término, $K*(D/Q)$, significa el costo de la puesta en marcha de la producción. Esto es, se debe poner en marcha la producción D/Q veces en el período de tiempo.

El tercer término, $h*(Q/2)$, significa el costo de la cantidad promedio a mantener en inventario durante un ciclo.

Es esta función de costos totales la que se desea minimizar. Por lo tanto se deriva respecto a Q, obteniendo:

$$0 = -\frac{DK}{Q^2} + \frac{h}{2} \quad (2)$$

Luego despejando Q se obtiene el Q^*, valor óptimo deseado:

$$Q^* = \sqrt{\frac{2DK}{h}} \quad (3)$$

4.2. Problema del Tamaño del Lote Económico (ELSP)

Este modelo es una generalización del modelo EOQ [10]. Intenta determinar las cantidades a producir en cada período para satisfacer los requerimientos de la demanda de un solo artículo, con demanda conocida (determinística), pudiendo ser diferente para cada período (dinámica), y con revisión periódica o continua de nivel de inventario. Asume que la demanda es satisfecha al inicio de cada período. La capacidad de producción y almacenamiento se supone ilimitada. No se permite backlogging, es decir que la demanda debe ser satisfecha en cada período. En cada período se cuenta con: costos de producción y costos de mantenimiento en inventario.

Los componentes de este modelo matemático son:

- $D_t \geq 0$: demanda en el período t, con $1 \leq t \leq T$
- $c_t(x) \geq 0$: costo de producir una cantidad x durante el período t, con $1 \leq t \leq T$
- $h_t(y) \geq 0$: costo de mantener en inventario una cantidad y, desde el período $t-1$ hasta el período t, con $2 \leq t \leq T$

El modelo matemático para el problema es entonces:

$$mín \sum_{t=1}^{T} \{c_t(x_t) + h_t(y_t)\} \quad (4)$$

s.a:

$$y_1 = y_{T+1} = 0 \quad (5)$$
$$x_t \geq 0 \qquad 1 \leq t \leq T \quad (6)$$
$$y_t \geq 0 \qquad 1 \leq t \leq T \quad (7)$$
$$x_t + y_t = d_t + y_{t+1} \qquad 1 \leq t \leq T \quad (8)$$

Notar que el inventario inicial y final se suponen cero.

Este modelo fue propuesto por Wagner y Whitin (1958) [34], quienes lo resolvieron con un algoritmo basado en un enfoque de programación dinámica de $O(T^2)$, donde T es el largo del horizonte de planificación, bajo el supuesto de que los costos unitarios de producción y de almacenamiento se mantienen constantes en todos los períodos. Se demostró luego que este algoritmo también es válido para el caso general de funciones de costos cóncavas (Zangwill, 1968) [36].

El algoritmo se basa en la propiedad de inventario cero, donde se demuestra que hay una solución óptima para la cual la cantidad a producir en un período determinado es positiva si y sólo si el nivel de inventario es cero.

Como consecuencia de esto, se restringe a producir sólo cuando el nivel de inventario sea cero.

Más adelante, se propusieron otros algoritmos más rápidos que resuelven el problema ELSP: Federgruen y Tzur (1991) [7], Wagelmans et al. (1992) [33] y Aggarwal and Park (1993) [1] propusieron algoritmos de $O(TlogT)$ y $O(T)$ bajo el mismo supuesto de costos unitarios de producción constantes.

Luego Atamtürk y Küçükyavuz (2008) [2] propusieron un algoritmo de $O(T^2)$ para el problema ELSP más genérico donde se agregan límites de inventario y costos fijos de mantener en inventario.

El ELSP no cuenta con restricciones de capacidad. En la realidad, no contar con este tipo de restricciones no es muy realista, ya sea por costos, espacio en inventarios, etc. Existen otros modelos bajo el problema CSILSP (Capacitated Single Item Lot Sizing Problem) con diferentes extensiones [4], que sí manejan esta realidad. Pero esto no estará bajo nuestro foco.

En Guner-Goren et al. (2010) [14] se presenta una amplía revisión de los trabajos que se han realizado con respecto al problema de ELSP. En el mismo, se explica el concepto de EOQ y se va apuntando hacia el problema de ELSP contemplando sus diferentes características (siempre basándose en la literatura revisada por los autores), como por ejemplo, demanda determinística y estática, horizonte de planificación finito e infinito, múltiples artículos y para un solo artículo, capacidad limitada e ilimitada, entre otras.

La metaheurística que se estudia para la resolución del problema está basada en algoritmos evolutivos. Se definen cada uno de los aspectos del problema que se deben considerar en el algoritmo evolutivo tales como número de niveles de producción (cantidad de artículos dependientes entre sí, que se producen en una misma planta), restricción de capacidad, tiempos de configuración, horizonte de planificación, tipo de demanda y permisión de escasez.

También se estudia y comparan las diferentes soluciones de cada uno de los trabajos revisados. Se detallan cada una de las características y los parámetros empleados en los algoritmos evolutivos implementados, tales como población inicial, probabilidad en la mutación y cruzamiento, técnica de selección, entre otros.

Finalmente, en este trabajo se concluye que las soluciones basadas en algoritmos evolutivos son más eficientes y rápidas que las técnicas enfocadas a soluciones exactas y otras heurísticas. También se concluye que la calidad de la solución usando algoritmos evolutivos depende fuertemente de los parámetros utilizados.

4.3. Problema del Tamaño del Lote Económico con Remanufacturación (ELSR)

El ELSR es una extensión del ELSP con retornos, donde los productos que son devueltos pueden ser remanufacturados.

El objetivo es determinar cuánto producir o remanufacturar en cada período, para cubrir la demanda a tiempo (sin backlogging) con productos nuevos o remanufacturados respectivamente, minimizando los costos involucrados.

Se mantienen las características mencionadas en el modelo ELSP, agregando costos de inventario para retornos, y que se conoce la cantidad de retornos en cada período.

Se han estudiado diferentes variantes del problema ELSR. Richter & Sombrutzki (2000) [27], amplían el modelo clásico de Wagner-Whitin al introducir el proceso de remanufactura: los productos regresan al productor y son almacenados para que posteriormente sean remanufacturados o desechados. Se manejan dos inventarios diferentes: uno para los retornos, y otro para los productos remanufacturados y productos nuevos.

Los modelos presentados en éste trabajo resuelven el problema de un solo artículo. Primero presentan un modelo con remanufacturación pura (los retornos remanufacturados son suficientes para satisfacer la demanda) y luego otro con remanufacturación alternativa (los remanufactura y la manufactura son necesarios para satisfacer la demanda).

En la Figura 1 se representa el flujo de artículos usados, para el problema de remanufacturación pura:

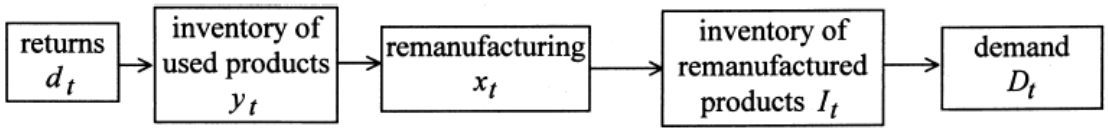


Figura 1: Flujo de artículos usados.

Con los siguientes componentes:

- D_t: representa la demanda de para un artículo en el período *t*.
- d_t: representa la cantidad de retornos conocida al principio de cada período *t*.
- I_t: representa el nivel de inventario de productos remanufacturados en el período *t*.
- H_t: representa el costo de mantener en inventario un producto remanufacturado en el período *t*.
- y_t: representa el nivel de inventario de retornos en el período *t*.
- h_t: representa el costo de mantener en inventario un retorno en el perído *t*.
- r_t: representa el costo de configuración o puesta en marcha de la actividad de remanufacturación en el período *t*.

A continuación, el modelo matemático, en dónde las ecuaciones de balance (11) y (12) describen el proceso de inventario para los productos remanufacturados y para los productos retornados respectivamente.

$$mín \ \sum_{t=1}^{T}(r_t * sign\ x_t \ + \ h_t * y_t + H_t * I_t \ ) \qquad (9)$$

s.a:

$$y_0 = I_0 = 0 \quad (10)$$
$$y_t = y_{t-1} - x_t + d_t, \quad (11)$$
$$I_t = I_{t-1} + x_t - D_t, \quad (12)$$
$$I_t, x_t, y_t \geq 0, \quad (13)$$
$$t = 1,2,\dots,T \quad (14)$$

Luego, los autores extienden el modelo agregando la opción alternativa de producción. La demanda puede ser satisfecha por productos manufacturados o remanufacturados que tendrán el mismo valor para el cliente. En la Figura 2 se puede apreciar el flujo de artículos extendido:

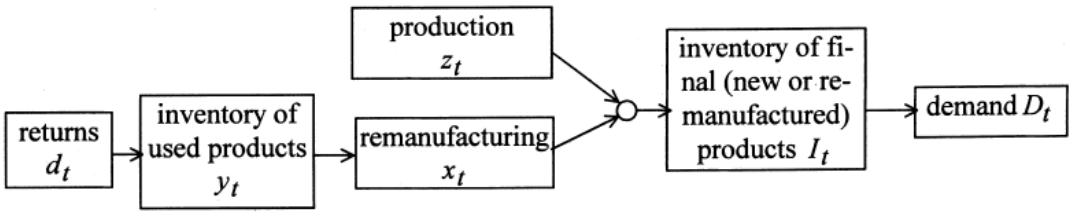


Figura 2: Flujo de artículos usados y producción.

Los nuevos componentes de este modelo son:

- s_t: representa el costo de configuración para la actividad de manufacturación en el período *t*.
- z_t: representa la cantidad a producir en el período *t*.

Para simplificar el problema, se supone igual el costo de mantener en inventario productos nuevos y remanufacturados. A continuación se presenta el modelo matemático:

$$mín \sum_{t=1}^{T} (r_t * sign\, x_t + s_t * sign\, z_t + h_t * y_t + H_t * I_t) \quad (15)$$

s.a:

$$y_0 = I_0 = 0, \quad (16)$$
$$y_t = y_{t-1} - x_t + d_t, \quad (17)$$
$$I_t = I_{t-1} + x_t + z_t - D_t, \quad (18)$$
$$I_t, x_t, y_t \geq 0, \quad (19)$$
$$t = 1,2,\dots,T \quad (20)$$

También demostraron que existe una solución óptima para la política de inventario cero y propusieron un algoritmo de programación dinámica para determinar los períodos para producir o remanufacturar.

Teunter et al. (2006) [32] estudian el modelo ELSR con tarea de producción y remanufacturación para satisfacer la demanda. La demanda y la cantidad de retornos se supone conocida para cada período, o al menos se puede pronosticar, siguen una distribución estacionaria. El modelo solo incluye costos de configuración y costos de mantenimiento en inventario. Los costos variables de producción y remanufacturación no se incluyen en el modelo ya que se suponen estacionarios, por lo tanto a largo plazo, estos costos no afectan sobre las decisiones de tamaño de lotes.

Se consideran dos casos para el costo fijo de configuración (set-up): conjunta o independiente para manufacturación y remanufacturación. El primer caso es adecuado si las operaciones de manufacturación y remanufacturación se llevan a cabo en la misma línea de producción. El segundo caso es para cuando se manejan líneas de producción separadas.

Los componentes del modelo de costos de configuración conjunta son:

- T: representa el horizonte de planificación.
- R_t: representa la cantidad de retornos disponibles en el período t.
- D_t: representa la demanda de artículos en el período t.
- x_t^r: representa la cantidad de artículos remanufacturados en el período t.
- x_t^m: representa la cantidad de artículos manufacturados en el período t.
- I_t^r: representa el nivel de inventario de retornos al final del período t.
- I_t^s: representa el nivel de inventario de artículos listos al final del período t.
- h_r: representa el costo de mantener en inventario un retorno en el perído t.
- h_s: representa el costo de mantener en inventario un artículo listo en el perído t.
- K: representa el costo de configuración conjunta.
- δ_t: 0 o 1, indicador para el costo de configuración en el período t.
- M: valor entero.

El modelo matemático es:

$$mín \sum_{t=1}^{T}(K\delta_t + h^r I_t^r + h^s I_t^s) \quad (21)$$

s.a:

$$I_{t-1}^r + R_t - x_t^r = I_t^r, \quad \forall\, t = 1, \dots, T \quad (22)$$
$$I_{t-1}^s + x_t^r + x_t^m - D_t = I_t^s, \quad \forall\, t = 1, \dots, T \quad (23)$$
$$M_t\delta_t \geq x_t^r + x_t^m, \quad \forall\, t = 1, \dots, T \quad (24)$$
$$\delta_t \in \{0, 1\},\ x_t^r,\ x_t^m,\ I_t^r,\ I_t^s \geq 0, \quad (25)$$
$$M_t = \sum_1^T D_i, \quad \forall\, t = 1, \dots, T \quad (26)$$

En este trabajo se generaliza el algoritmo de Wagner-Whitin. Para el caso del costo fijo de configuración conjunta, proporcionan un algoritmo exacto de programación dinámica en tiempo polinomial de $O(T^4)$. También estudiaron y compararon el desempeño computacional de versiones modificadas de las heurísticas Silver Meal (SM), Least Unit Cost (LUC) y Part Period Balancing (PPB) [31] para sistemas con manufacturación y remanufacturación. Las tres heurísticas se centran únicamente en la demanda inmediata sin tomar en cuenta los costos asociados a las demandas futuras. Además, sólo consideran soluciones que satisfacen la propiedad de inventario cero. La heurística SM

elige el orden que minimice el costo por período, la heurística LUC elige el orden que minimice el costo por unidad ordenada y la heurística PPB elige el tamaño del lote que minimiza la diferencia entre el costo de configuración y el costo total de almacenamiento.

En Schulz (2011) [29] se hace una generalización de la heurística basada en Silver-Meal introducida por Teunter [32], para costos de configuración separados para producción y remanufacturación, sin disposición final ni restricción en las capacidades, logrando una mejora en el desempeño de la misma.

Se estudian los siguientes casos:

- ✓ Opción 1: Sólo manufacturación (la demanda es satisfecha con elementos manufacturados).
- ✓ Opción 2: Remanufacturación, y manufacturación si fuera necesario (la demanda es satisfecha con retornos remanufacturados. Si esto no fuera suficiente, también se usan elementos manufacturados en conjunto con los remanufacturados).
- ✓ Opción 3: Manufacturación primero, remanufacturación después (la demanda se satisface con una cantidad específica de elementos manufacturados. Cuando estos se terminan, se pasa a utilizar los retornos remanufacturados).
- ✓ Opción 4: Remanufacturación primero, manufacturación después (la demanda se satisface con una cantidad específica de retornos remanufacturados. Cuando estos se terminan, se pasa a utilizar elementos manufacturados).

Las mejoras que se presentan sobre el algoritmo Silver-Meal son:

- Buscar disminuir los costos al combinar dos o más ventanas o bloques consecutivos.
- Búsqueda del decremento del costo total al incrementar la remanufacturación de un periodo, y al mismo tiempo decrementar la misma cantidad en la manufacturación en el siguiente período.

En Baki et al. (2014) [3] se estudia el problema de manufacturación y remanufacturación de retornos en un horizonte de planeación finito (N), poniendo foco en un importante aspecto táctico de fabricación para su reutilización.

La cantidad de retornos varía en cada período. Los mismos pueden ser remanufacturados inmediatamente o pueden quedar en inventario para su posterior remanufacturación (esto tiene un costo asociado). La demanda es determinística y se manejan costos fijos para preparación de producción y remanufacturación.

La cantidad manufacturada en un período puede ser utilizada para satisfacer toda la demanda o parte de la misma, lo mismo aplica a la remanufacturación de un período. Agregan dos períodos ficticios (0 y N+1) en los que la demanda es cero y por tanto no hay actividad manufacturera.

Como contribución a la literatura existente, trabajan sobre la formulación de un modelo MILP (Mixed Integer Linear Programing) alternativo al modelo clásico presentado en [27] y [32].

Los componentes del modelo matemático son:

- N: horizonte de planeación
- D_i: número de productos demandados en el período *i*
- R_i: número de productos retornados al comienzo del período *i*
- K^S: costo unitario de configuración para manufacturación
- K^R: costo unitario de configuración para remanufacturación
- h^S: costo de mantener en inventario un producto listo del período i al período i+1
- h^R: costo de mantener en inventario un retorno del período i al período i+1
- d_{ij}: demanda acumulada entre el período i y el período j
- r_{ij}: retornos acumulados entre el período i y el período j
- Q_i^S: cantidad de productos manufacturados en el período i
- Q_i^R: cantidad de productos remanufacturados en el período i
- I_i^S: inventario de productos listos al finalizar el período i
- I_i^R: inventario de retornos al finalizar el período i
- y_i^S: variable binaria: 1 si nuevos productos son manufacturados en el período i, 0 en otro caso
- y_i^R: variable binaria: 1 si retornos son remanufacturados en el período i, 0 en otro caso
- x_{ij}^S: variable binaria: 1 si ocurre manufacturación en el período i y el próximo período es (j+1), 0 en otro caso
- x_{ij}^R: variable binaria: 1 si ocurre remanufacturación en el período i y el próximo período es (j+1), 0 en otro caso

El modelo matemático es:

$$min\ h^S \sum_{i=1}^{N} I_i^S + h^R \sum_{i=1}^{N} I_i^R + K^S \sum_{i=1}^{N} \sum_{j=i}^{N} x_{ij}^S + K^R \sum_{i=1}^{N} \sum_{j=i}^{N} x_{ij}^R \qquad (27)$$

s.a:

$$I_i^S = I_{i-1}^S + Q_i^S + Q_i^R - D_i \qquad \forall\ i = 1, \dots, N \qquad (28)$$
$$I_i^R = I_{i-1}^R + R_i - Q_i^R \qquad \forall\ i = 1, \dots, N \qquad (29)$$
$$Q_i^S \leq \sum_{j=i}^{N} d_{ij} x_{ij}^S \qquad \forall\ i = 1, \dots, N \qquad (30)$$
$$Q_i^R \leq \sum_{j=i}^{N} d_{ij} x_{ij}^R \qquad \forall\ i = 1, \dots, N \qquad (31)$$
$$\sum_{j=0}^{N} x_{0j}^S = 1 \qquad (32)$$
$$\sum_{i=0}^{j} x_{ij}^S = \sum_{k=j+1}^{N} x_{(j+1)k}^S \qquad \forall\ j = 1, \dots, N-1 \qquad (33)$$
$$\sum_{i=0}^{N} x_{iN}^S = 1 \qquad (34)$$
$$\sum_{j=0}^{N} x_{0j}^R = 1 \qquad (35)$$
$$\sum_{i=0}^{j} x_{ij}^R = \sum_{k=j+1}^{N} x_{(j+1)k}^R \qquad \forall\ j = 1, \dots, N-1 \qquad (36)$$
$$\sum_{j=0}^{N} x_{jN}^R = 1 \qquad (37)$$
$$x_{ij}^S, x_{ij}^R \in \{0,1\}, Q_i^S, Q_i^R, I_i^S, I_i^R \geq 0 \quad \forall\ i = 0,1, \dots, N\ y\ \forall\ j = 0,1, \dots, N \qquad (38)$$

Muestran que el modelo presentado tiene una ventaja importante frente al modelo clásico: las cotas inferiores son en general mucho mejores, cuando las restricciones de integralidad están relajadas.

Presentan una heurística eficiente que hace uso de programación dinámica y del algoritmo de Wagner-Whithin para resolver el problema.

La heurística se basa en la observación de que una solución factible del modelo se puede descomponer en una serie de bloques con distintos patrones de configuración para la manufacturación y la remanufacturación. Cada bloque (s, t) costa de un conjunto de períodos consecutivos (s, s+1,…, t), tal que contienen al menos un período en el que ocurre manufacturación o remanufacturación. Si existe un período i en el que ocurre manufacturación y un período j en el que ocurre remanufacturación, dentro del mismo bloque, entonces $i \leq j$. El problema se reduce a encontrar la cadena óptima de bloques.

Li et al. (2014) [17] trabaja sobre el mismo modelo matemático que Teunter [32], que asume demanda y cantidad de retornos conocidos en cada período, donde los costos que participan son los de configuración y almacenamiento en inventario. Además los inventarios iniciales, para productos listos y retornos, son cero.

El estudio introduce un sofisticado procedimiento Tabu Search para el modelo ELSR con costos fijos, que produce una solución de alta calidad. Se divide el horizonte de planificación en una cadena de bloques, en donde cada bloque representa un conjunto de períodos con una secuencia de fabricación, remanufacturación o ambos. Para hallar el costo de cada bloque, se utiliza un modelo de programación lineal (LP) que puede ser resuelto de forma rápida y eficiente por diferentes programas (solvers). Luego, dado que el costo de cada bloque es conocido, se haya una solución inicial que contenga la mejor combinación de bloques, tal que minimice el costo total de producción, remanufacturación y almacenamiento en inventario, para esto se utiliza el algoritmo del camino más corto. A través de cuatro operadores (one-shift, two-shift, exchange, y cross two-shift), se exploran los alrededores de la solución, obteniendo soluciones vecinas. Para mantener diversidad en el espacio de búsqueda, se cuenta con una estrategia que obliga a buscar en zonas que aún no han sido exploradas cuando, luego de varias iteraciones consecutivas, no se logra una mejora en la solución obtenida.

Este procedimiento produjo una solución óptima en el 96.60% de los problemas evaluados.

En Sifaleras et al. (2015) [30] se trabaja con un modelo donde el horizonte de planificación es finito. Además, la demanda es determinística, no estacionaria, y puede ser satisfecha tanto con productos manufacturados como remanufacturados, los cuales no presentan diferencia en la calidad. También se asume como conocida, y no estacionaria, la cantidad de retornos para cada uno de los períodos. El modelo estudiado es el mismo que el planteado en Teunter [32].

Se trabaja sobre la metaheurística Variable Neighborhood Search (VNS), algoritmo para encontrar una solución de buena calidad para el ELSR, y que se

basa en recorrer diferentes vecindades, explorando las mejores soluciones de cada una de ellas.

Sobre esta técnica se proponen dos variantes: OGVNS (Ordered Variable Neighborhood Descent) y RGVNS (Randomised Variable Neighborhood Descent). En ambas variantes, el algoritmo VNS constituye la intensificación, y la fase de Shaking constituye la diversificación. Donde el objetivo de esta última es escapar de óptimos locales, permitiendo evaluar áreas inexploradas de la región factible. Cuando el algoritmo VNS no logra mejorar la solución actual por medio de Búsqueda Local, comienza la fase de Shaking para explorar vecindades mayores.

La variante OGVNS, se basa en la cantidad de mejoras locales por vecindad, de esta manera, los movimientos más usados se consideran primero.

La variante RGVNS, usa un método local de búsqueda aleatoria, y un nuevo tipo de Shaking. Esta fase de Shaking consiste en mezclar el orden de los barrios, para obtener una solución nueva aleatoria. Como variante, se implementó la posibilidad de mezclar k barrios consecutivos, lo cual permite explorar regiones más lejanas que contienen soluciones cercanas al óptimo.

El objetivo del modelo es determinar el número de elementos remanufacturados y manufacturados en cada período, de tal forma de minimizar la suma de los costos de ambos procesos (manufacturación y remanufacturación), y el costo de inventario de retornos y de elementos listos bajo ciertas restricciones.

Si bien este modelo es muy similar al presentado en [32], se manejan dos costos de configuración diferentes, uno para la manufacturación y otro para la remanufacturación. Mientras que en [32], se maneja un costo de configuración conjunta para ambos procesos.

Además del juego de datos presentado en Schulz (2011) [29], el cual consiste en de 6480 instancias, se ha utilizado un nuevo conjunto de datos para poder realizar pruebas y demostrar la robustez de este algoritmo. Dicho conjunto de datos es cuatro veces más grande que el de Schulz, y cada instancia presenta 52 períodos.

La misma mostró que la metaheurística VNS es capaz de alcanzar una diferencia con respecto a la solución óptima de tan sólo el 0,283% en promedio, en un tiempo de 8.3 segundos aproximadamente.

En Piñeyro & Viera (2015) [26] se considera la formulación del ELSR presentada por Teunter et al. [32], agregando las siguientes contribuciones: se presenta un nuevo resultado teórico sobre la forma de la solución óptima del ELSR, con el mismo se mejora el algoritmo Tabu Search presentado en Piñeyro y Viera [25] para el ELSR, y finalmente se evalúa la versión original del algoritmo y la versión actualizada, con instancias propuestas por Sifaleras et al. [30].

Como resultado se llega a que los dos algoritmos (el original y el optimizado) superan, en más de 90% de los casos, el procedimiento basado en VNS propuesto en [30], y en aproximadamente la décima parte del tiempo de cálculo en el peor de los casos.

4.3.1. ELSR con disposición final

Richter & Weber (2001) [28] extendieron el modelo presentado en [27] introduciendo costos variables (dinámicos) de producción y remanufacturación, y la opción de desechar o disponer en cada período determinada cantidad de retornos que no serán remanufacturados. La Figura 3 representa el flujo de artículos para éste modelo:

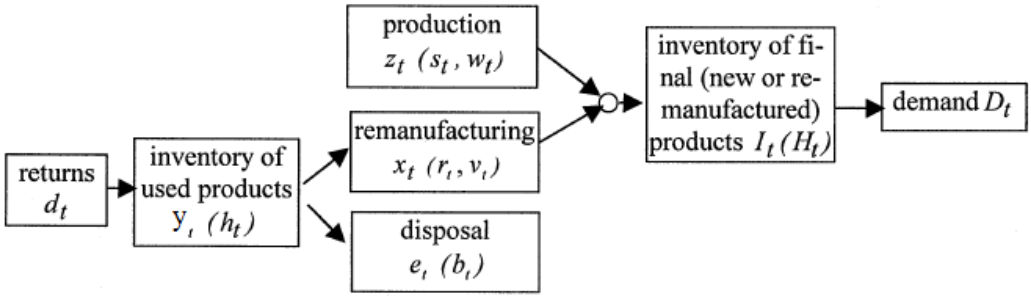


Figura 3: Flujo de artículos usados con disposición y producción.

Los nuevos componentes en el modelo son:

- w_t: representa el costo variable de producción en el período *t*.
- v_t: representa el costo variable de remanufacturación en el período *t*.
- e_t: representa la cantidad a disponer o desechar en cada período *t*.
- b_t: representa el costo unitario de lo que se desecha o dispone en el período *t*. Este costo puede ser negativo, generando una ganancia, en el caso donde el retorno es vendido.

Suponiendo que los costos de disposición final son no decrecientes, se intenta no aplazar esta actividad. Esto lleva a que la disposición final se realice tan pronto como se pueda.

Se resuelve para el caso particular donde la cantidad inicial de retornos es muy grande, entonces, por lo anterior, se maneja una sola tarea de disposición final en el primer período.

El modelo matemático es el siguiente:

$$mín \sum_{t=1}^{T} \begin{pmatrix} r_t * sign\, x_t + v_t * x_t + s_t * sign\, z_t + \\ w_t * z_t + h_t * y_t + H_t * I_t \end{pmatrix} + b * e \qquad (39)$$

s.a:

$$y_0 = I_0 = 0, \qquad (40)$$
$$y_t = y_{t-1} - x_t, \quad t = 2, \dots, T, \qquad (41)$$
$$I_t = I_{t-1} + x_t + z_t - D_t, \quad t = 1, \dots, T, \qquad (42)$$
$$y_1 = d - e - x_1, \qquad (43)$$
$$I_t, x_t, y_t, z_t \geq 0, \quad t = 1, \dots, T \qquad (44)$$

Una variante del ELSR estudiada en Golany et al. (2001) [12] fue la de manejar costos generales cóncavos, demostrando que este también es un problema NP-hard. Este modelo también maneja disposición final. Las opciones

posibles sobre los retornos son: eliminar lo que se considera desecho, mantenerlos en inventario para futuras decisiones, remanufacturarlos.

Los componentes de este modelo son:

- B_t: representa la cantidad de retornos disponibles en el período *t*.
- D_t: representa la demanda de artículos nuevos en el período *t*.
- x_t: representa la cantidad de artículos producidos en el período *t*.
- y_t: representa la cantidad en inventario de artículos nuevos al final del período *t* (y_0 y y_T son dados).
- z_t: representa la cantidad de retornos que se remanufacturan en el período *t*.
- u_t: representa la cantidad en inventario de retornos al final del período *t* (u_0 y u_T son dados).
- v_t: representa la cantidad de retornos desechados en el período *t*.
- $P_t(x_t) \geq 0$: representa el costo de producción en el período *t*.
- $H_t(y_t) \geq 0$: representa el costo de mantener en inventario un artículo nuevo en el período *t.*
- $R_t(z_t) \geq 0$: representa el costo de remanufacturación en el período *t*.
- $W_t(u_t) \geq 0$: representa el costo de mantener en inventario un retorno en el período *t.*
- $S_t(v_t) \geq 0$: representa el costo de desechar en el período *t*.

Por simplicidad y sin pérdida de generalidad se consideran

$$P_t(0) = H_t(0) = R_t(0) = W_t(0) = S_t(0) = 0.$$

A continuación, el modelo matemático:

$$mín \sum_{t=1}^{T} P_t(x_t) + \sum_{t=1}^{T-1} H_t(y_t) + \sum_{t=1}^{T} R_t(z_t) + \sum_{t=1}^{T-1} W_t(u_t) + \sum_{t=1}^{T} S_t(v_t) \quad (45)$$

s.a:

$$x_t + y_{t-1} - y_t + z_t = D_t \qquad \forall t = 1, \dots, T \quad (46)$$

$$z_t + u_t - u_{t-1} + v_t = B_t \qquad \forall t = 1, \dots, T \quad (47)$$

$$x_t, y_t, z_t, u_t, v_t \geq 0, \qquad \forall t = 1, \dots, T \quad (48)$$

Yang et al. (2005) [35] trabajan sobre el mismo escenario que en [12]. Modelan el problema con una formulación del tipo de flujo de red. Presentan un algoritmo de programación dinámica para la resolución del problema y demuestran que el problema de costos cóncavos es NP-hard, incluso cuando todas las funciones de costo son estacionarias. Además, se desarrolla una heurística en tiempo polinomial de $O(T^4)$ para la resolución del problema con función de costos cóncavas.

Los componentes de este modelo matemático son:

- B_t: representa la cantidad de retornos disponibles en el período *t*.
- D_t: representa la demanda de artículos nuevos en el período *t*.
- x_t: representa la cantidad de artículos producidos en el período *t*.

- y_t: representa la cantidad en inventario de artículos nuevos al final del período *t*, con t de *1* a *T-1*.
- z_t: representa la cantidad de retornos que se remanufacturan en el período *t*.
- u_t: representa la cantidad en inventario de retornos al final del período *t*, con t de *1* a *T-1*.
- v_t: representa la cantidad de retornos desechados en el período *t*.
- U_0: valor inicial dado de nivel de inventario de retornos.
- U_t: valor final dado de nivel de inventario de retornos.
- Y_0: valor inicial dado de nivel de inventario de productos nuevos.
- Y_T: valor final dado de nivel de inventario de productos nuevos.
- $P_t(x_t) \geq 0$: representa el costo de producción en el período *t*.
- $H_t(y_t) \geq 0$: representa el costo de mantener en inventario un artículo nuevo en el período *t*, con t de *1* a *T-1*.
- $R_t(z_t) \geq 0$: representa el costo de remanufacturación en el período *t*.
- $W_t(u_t) \geq 0$: representa el costo de mantener en inventario un retorno en el período *t*, con t de *1* a *T-1*.
- $S_t(v_t) \geq 0$: representa el costo de desechar en el período *t*.

La función objetivo es igual a la modelada en [12] pero se agregan más restricciones sobre las ecuaciones de balance que involucran los niveles de inventario inicial y final de retornos y productos nuevos.

El modelo matemático es:

$$mín \sum_{t=1}^{T} P_t(x_t) + \sum_{t=1}^{T-1} H_t(y_t) + \sum_{t=1}^{T} R_t(z_t) + \sum_{t=1}^{T-1} W_t(u_t) + \sum_{t=1}^{T} S_t(v_t) \quad (49)$$

s.a:

$$z_1 - U_0 + u_1 + v_1 = B_1 \quad (50)$$
$$z_t + u_t - u_{t-1} + v_t = B_t \quad \forall t = 2, \dots, T-1 \quad (51)$$
$$z_T + U_T - u_{T-1} + v_T = B_T \quad (52)$$
$$z_1 + x_1 + Y_0 - y_1 = D_1 \quad (53)$$
$$z_t + x_t + y_{t-1} - y_t = D_t \quad \forall t = 2, \dots, T-1 \quad (54)$$
$$z_T + x_T + y_{T-1} - Y_T = D_T \quad (55)$$
$$x_t, z_t, v_t \geq 0, \quad \forall t = 1, \dots, T \quad (56)$$
$$y_t, u_t \geq 0, \quad \forall t = 1, \dots, T-1 \quad (57)$$

Piñeyro & Viera (2009) [25] estudian también el modelo ELSR con producción y disposición final. Al igual que los otros trabajos mencionados antes en esta sección, manejan costos separados de producción, remanufacturación y disposición final, además de costos de almacenamiento para productos listos (manufacturados y remanufacturados) y retornos.

La Figura 4 representa el flujo de artículos para éste modelo:

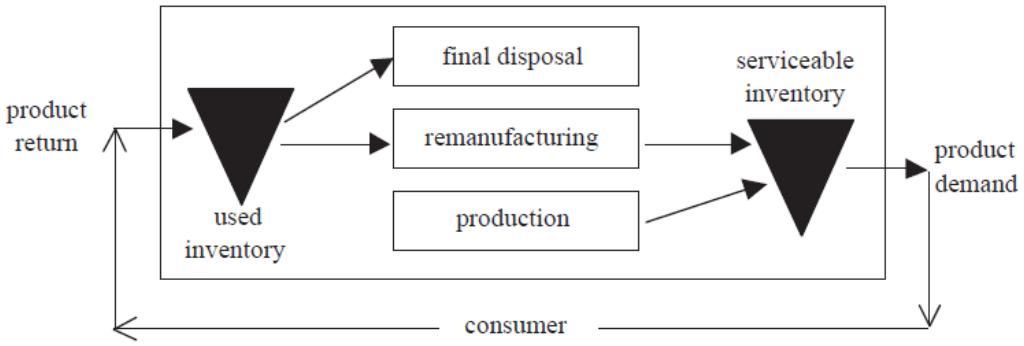


Figura 4: Flujo de artículos usados con disposición y producción.

Los componentes del modelo matemático son:

- $T > 0$: horizonte de planeación, con T $< +\infty$
- $D_t \geq 0$: demanda en el período *t*, con $1 \leq t \leq T$
- $D_{ij} \geq 0$: demanda acumulada entre los períodos i y j, con $1 \leq i \leq j \leq T$
- $R_t \geq 0$: cantidad de productos usados que retornan en el período *t*, con $1 \leq t \leq T$
- $R_{ij} \geq 0$: cantidad acumulada de retornos entre los períodos *i* y *j*, con $1 \leq i \leq j \leq T$
- $p_t \geq 0$: cantidad de productos nuevos producidos en el período *t*, con $1 \leq t \leq T$
- $r_t \geq 0$: cantidad de productos remanufacturados en el período *t*, con $1 \leq t \leq T$
- $r_{ij} \geq 0$: cantidad acumulada de productos remanufacturados entre los períodos *i* y *j*, con $1 \leq i \leq j \leq T$
- $d_t \geq 0$: cantidad de productos desechados en el período *t*, con $1 \leq t \leq T$
- $y_t^s \geq 0$: inventario de productos remanufacturados y nuevos (listos) en el período *t*, con $1 \leq t \leq T$
- $y_t^r \geq 0$: inventario de productos usados en el período *t*, con $1 \leq t \leq T$
- $K_t^p \geq 0$: costo de preparación para producción en el período *t*, con $1 \leq t \leq T$
- $K_t^r \geq 0$: costo de preparación para remanufacturación en el período *t*, con $1 \leq t \leq T$
- $K_t^d \geq 0$: costo de preparación para disposición final en el período *t*, con $1 \leq t \leq T$
- $c_t^p \geq 0$: costo unitario de producir en el período *t*, con $1 \leq t \leq T$
- $c_t^r \geq 0$: costo unitario de remanufacturar en el período *t*, con $1 \leq t \leq T$
- $c_t^d \geq 0$: costo unitario de disposición final en el período *t*, con $1 \leq t \leq T$
- $h_t^s \geq 0$: costo unitario de mantener en inventario un producto listo en el período *t*, con $1 \leq t \leq T$
- $h_t^r \geq 0$: costo unitario de mantener en inventario un retorno en el período *t*, con $1 \leq t \leq T$

El modelo matemático es:

$$min \sum_{t=1}^{T} \left\{ \begin{matrix} K_t^p \delta_t^p + c_t^p p_t + K_t^r \delta_t^r + c_t^r r_t + K_t^d \delta_t^d + \\ c_t^d d_t + h_t^s y_t^s + h_t^r y_t^r \end{matrix} \right\} \quad (58)$$

s.a:

$$y_t^s = y_{t-1}^s + p_t + r_t - D_t \qquad \forall\, t = 1, \dots, T \qquad (59)$$
$$y_t^r = y_{t-1}^r + R_t - r_t - d_t \qquad \forall\, t = 1, \dots, T \qquad (60)$$
$$M\delta_t^p \geq p_t \qquad \forall\, t = 1, \dots, T \qquad (61)$$
$$M\delta_t^r \geq r_t \qquad \forall\, t = 1, \dots, T \qquad (62)$$
$$M\delta_t^d \geq d_t \qquad \forall\, t = 1, \dots, T \qquad (63)$$
$$y_0^s = y_0^r = 0 \qquad \forall\, t = 1, \dots, T \qquad (64)$$
$$\delta_t^p, \delta_t^r, \delta_t^d \in \{0, 1\} \qquad \forall\, t = 1, \dots, T \qquad (65)$$
$$p_t, r_t, d_t, y_t^r, y_t^s \geq 0 \qquad \forall\, t = 1, \dots, T \qquad (66)$$

Se presentan un conjunto de políticas (que se centran en la actividad de remanufactura) para resolver éste tipo de modelo, que son eficaces respecto al costo y al tiempo. Las mismas se basan en el concepto de "divide y vencerás", ya que descomponen el problema en los subproblemas de producción, remanufacturación y disposición final, y los resuelve por separado.

Los problemas de producción y disposición final pueden ser resueltos de manera óptima por separado en $O(T^2)$ usando el algoritmo de Wagner-Whitin. La solución óptima para el ELSR se reduce entonces a encontrar el plan de remanufactura que permite determinar la producción óptima y planes de disposición final de manera eficaz. Se introduce una regla simple para determinar la cantidad a remanufacturar en un determinado período: es el mínimo entre los retornos disponibles y la demanda acumulada entre el período actual y el próximo período con remanufactura.

Algunas de las políticas presentadas en el trabajo son:

- ✓ P1RM: se produce únicamente en el primer período. La cantidad a producir se determina para asegurar que en el resto de los períodos (M) la demanda pueda ser satisfecha con la remanufacturación.
- ✓ PNRM: es una generalización de la política anterior, pero se produce lo necesario en los N primeros períodos.
- ✓ Rp2p: en cada período se trata de cubrir la demanda con la remanufacturación. En caso de que los retornos disponibles no sean suficientes, se recurre a la producción.
- ✓ FP4R: los períodos en que se realizará remanufacturación son fijados. La cantidad a remanufacturar es determinada como el valor mínimo entre los retornos disponibles en el período actual, y la demanda acumulada hasta el próximo período de remanufacturación.

También se proporciona un procedimiento Basic Tabu Search (BTS) con el fin de obtener una solución de buena calidad para el ELSR. El mismo utiliza la regla simple antes mencionada y recibe como parámetro los períodos en que se permite remanufacturación. El metaheurístico de Tabu Search es un proceso iterativo basado en la exploración de la información almacenada en estructuras en memoria.

En Piñeyro (2013) [24] se extiende la investigación del trabajo [25]. Uno de los principales objetivos de ésta investigación es confirmar la importancia de la regla simple presentada en [25], para determinar las cantidades a remanufacturar y su importancia en el éxito del procedimiento BTS.

Suponiendo que: los períodos en los que se permite la remanufacturación se conocen de antemano, las cantidades a remanufacturar son positivas y es rentable la remanufacturación, demuestran que el plan de remanufacturación de costo perfecto se obtiene en un tiempo lineal. Para el caso particular de un solo período, demuestran que la cantidad óptima de remanufacturación se puede determinar en a lo sumo tiempo $O(T^3)$.

En Parsopoulos et al. (2015) [23] se investiga el potencial del algoritmo Differential Evolution (DE), basado en población. Una población contiene individuos, con ellos es posible sondear de forma iterativa en el espacio de búsqueda, obteniendo nuevos puntos o individuos que sean más próximos a la solución. Se estudian diferentes combinaciones en los parámetros de configuración de este algoritmo.

En este trabajo se menciona que recientemente se ha demostrado que éste algoritmo (DE) es superior que un Algoritmo Genético (AG), otro algoritmo del mismo tipo. Los operadores en (DE) están basados en vectores de diferencias, mientras que en (AG) se utilizan operadores binarios.

4.4. ELSR con diferentes niveles de calidad para los retornos

El ELSR con clasificación de los artículos usados y retornados extiende aún más los problemas anteriores. En este modelo, existe un paso previo en el cual los retornos son clasificados dependiendo del estado en el que llegan a la fábrica. Dado que un retorno más deteriorado tendrá un costo de remanufacturación mayor que otro menos deteriorado, es necesario considerar diferentes posibilidades a la hora de tratar cada uno de ellos. Esto puede llevar a variaciones en el modelo, ya que se introducen nuevas variables como el costo fijo de clasificación, o diferentes inventarios para retornos de diferentes calidades. Esto se acerca aún más a una situación real, ya que la discriminación en calidades afecta directamente al costo del proceso.

Existen estudios realizados por Ferguson (Ferguson et al. (2006)[8] y Ferguson et al. (2009)[9]) que indican que los costos totales descienden considerablemente si se tiene en cuenta clasificación en los retornos, frente a un modelo donde no existe dicha clasificación.

En [8], se considera un modelo con demanda determinística y dinámica. Dado que cada empresa conoce aproximadamente la vida útil de sus productos, se puede estimar la cantidad de retornos en cada período. Además, el horizonte de planificación es finito y los retornos que no se remanufacturan pueden ser vendidos, produciendo una ganancia (valor de rescate). Por otra parte, backlogging no está permitido (se permiten faltantes), haciendo que la demanda no satisfecha se convierta, de alguna forma, en perdida.

La revisión es periódica, lo que significa que las diferentes decisiones se toman siempre en el mismo momento del período, en este caso al comienzo de cada período. Las acciones son instantáneas (tiempo de entrega cero).

El costo de remanufacturación se representa a través de una función lineal, convexa e incremental a tramos, donde cada tramo *i* representa el costo unitario de remanufacturar elementos de calidad *i*. Como se puede apreciar en la Figura 5, todos los tramos son incrementales, pero a peor calidad *i* más inclinado es el tramo. Dado que los costos no dependen del período, los mismos son estacionarios.

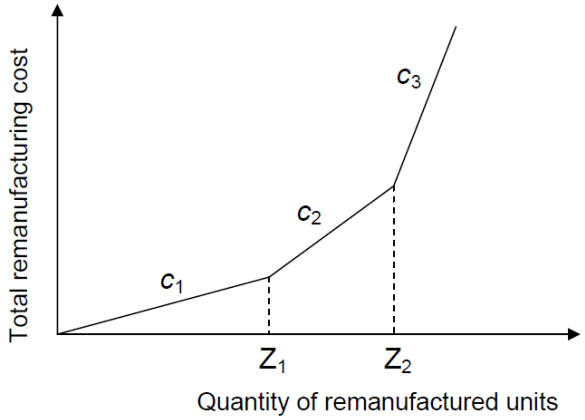


Figura 5: Curva de costos de remanufacturación

Por otra parte, los retornos que ya no pueden ser remanufacturados son vendidos por unidad, obteniendo la ganancia llamada valor de rescate.

Los componentes del modelo matemático son:

Índices:

i: Categoría de calidades, $i = 1$ (mejor),..., I (peor)
t: Períodos de tiempo, $t = 1,..., T$

Parámetros:

D_i: Demanda de productos remanufacturados en el período t
B_{it}: Cantidad de retornos de calidad i en el período t
s: Valor unitario de rescate de los retornos que no serán remanufacturados
h_i: Costo unitario por mantener en inventario de un retorno de calidad i
h_r: Costo unitario por mantener en inventario de un elemento remanufacturado
c_i: Costo unitario de remanufacturación de un retorno de calidad i

Variables de decisión:

z_{it}: Cantidad de retornos de calidad i remanufacturados en el período t
v_{it}: Cantidad de retornos que no serán remanufacturados, sino vendidos, de calidad i en el período t
y_t: Cantidad de productos remanufacturados en inventario al final del período t
u_{it}: Cantidad de retornos remanufacturados de calidad i en el período t

El modelo matemático propuesto es el siguiente:

$$min \sum_{t=1}^{T} \sum_{i=1}^{I} \{c_i z_{i,t} - sv_{i,t} + h_i u_{i,t}\} + h_r y_t \quad (67)$$

s.a:

$$y_{t-1} - y_t + \sum_{i=1}^{I} z_{i,t} = D_t \quad \forall\, t = 1, \dots, T, \quad (68)$$

$$z_{i,t} + u_{i,t} - u_{i,t-1} + v_{i,t} = B_{i,t} \quad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \quad (69)$$

$$z_{i,t}, u_{i,t}, v_{i,t}, y_t \geq 0 \quad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \quad (70)$$

La investigación numérica muestra que en un modelo con clasificación de retornos presenta una mejora con respecto al costo total del 11% (en promedio) frente a un sistema sin clasificación de retornos.

Se trabaja con una heurística de no inventario, cuyo resultado se compara contra la solución óptima. Dicha heurística asume que en cada período, la cantidad de retornos es lo suficientemente alta como para no tener la necesidad de manufacturar, ya que la demanda se satisface solo con elementos remanufacturados.

Esta política es fácil de implementar, ya que se van remanufacturando los retornos en orden de calidad, es decir, se remanufacturan los retornos de mejor a peor calidad, hasta satisfacer la demanda de dicho período.

Análisis numéricos demuestran que la solución obtenida con la heurística de no inventario es un 4% peor que la solución óptima.

En [9] se presenta con un modelo de revisión periódica, y con demanda determinística para productos remanufacturados. También se trabaja con un valor de rescate para los retornos que no se remanufacturan, y el backlogging está permitido.

La calidad de los retornos es representado por un número real entre 0 y 1 donde 0 significa que el retorno ya no puede ser remanufacturado y debe venderse, mientras que una calidad igual a 1 es la mejor calidad de un retorno. Si bien las empresas no saben a priori la calidad de cada retorno, se conoce una distribución de probabilidad, también entre 0 y 1, en donde las calidades son mapeadas.

Un retorno remanufacturado no es un reemplazo perfecto de un producto nuevo, pero como la demanda es satisfecha solo con remanufacturación, solo se cuenta con un inventario de artículos remanufacturados. En caso de que la demanda también se satisficiera con productos manufacturados, se manejarían inventarios separados para productos nuevos y productos remanufacturados.

Los retornos que no serán remanufacturados, pero serán vendidos, también son clasificados por su calidad y esto tiene un impacto en el precio de venta ya que este depende de las condiciones en las que esté el artículo. El precio de retorno varía con la calidad del producto a vender. A mayor calidad, mayor será el precio de venta.

Esta vez, el modelo presenta una función objetivo a maximizar, ya que representa la ganancia a lo largo de todo el horizonte de planificación.

Los componentes del modelo matemático son:

Índices:
i: Categoría de calidades, i=1 (peor), . . ., I (mejor)
t: Período de tiempo, t=1, . . ., T

Parámetros:
D_t: Demanda de productos remanufacturados en el período t
B_{it}: Cantidad de retornos de calidad i en el período t
c_{it}: Costo unitario de remanufacturación de un retorno de calidad i en el período t
s_{it}: Valor unitario de rescate de un retorno de calidad i en el período t
s_r: Valor unitario de rescate de un producto remanufacturado
h_{it}: Costo unitario por mantener en inventario de un retorno de calidad i en el período t
h_r: Costo unitario por mantener en inventario un producto remanufacturado
b: Costo unitario de backlogging por período
p_r: Precio unitario de venta de un producto remanufacturado

Variables de decisión:
z_{it}: Cantidad de retornos de calidad i remanufacturados en el período t
v_{it}: Cantidad de retornos de calidad i rescatados al final del período t

Variables de estado:
u_{it}: Cantidad de retornos de calidad i en el inventario al final del período t
y_t: Cantidad de productos remanufacturados en el inventario al final del período t
g_t: Total de demanda insatisfecha en el período t

Variables auxiliares:
U_t: Número total de retornos en inventarios al final del período t
Z_t: Número total de retornos remanufacturados en el período t

El modelo matemático propuesto es el siguiente:

$$max_{z_{it},v_{it}} TP = \sum_{t=1}^{T}\{ \sum_{i=1}^{I}\big((p_r - c_{it})z_{it}^{*} + s_{it}v_{it} - h_{it}u_{it}\big) + p_r(y_{t-1} - y_t) - h_r y_t - \lambda b g_t\} \quad (71)$$

s.a:

$$y_{t-1} - y_t + \sum_{i=1}^{I} z_{i,t} - \lambda g_{t-1} + g_t = D_t \quad \forall\, t = 1, \dots, T, \quad (72)$$

$$z_{i,t} + u_{i,t} - u_{i,t-1} + v_{i,t} = B_{i,t} \quad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \quad (73)$$

$$z_{i,t}, u_{i,t}, v_{i,t} \geq 0 \quad \forall\, t = 1, \dots, T, \quad \forall\, i = 1, \dots, I \quad (74)$$

$$y_t, g_t \geq 0 \quad \forall\, t = 1, \dots, T \quad (75)$$

Donde $z_{i,t}^{*}$ es la cantidad óptima de retornos de calidad i a remanufacturar en el período t, y se define como: $z_{i,t}^{*} = B_{i,t} - u_{i,t} + u_{i,t-1} - v_{i,t} \quad \forall\, t, i$

Este modelo presenta algunos costos estacionarios (h_r y b), como también costos que no son estacionarios (c_{it} y h_{it}).

Todos los resultados obtenidos en [9] asumen que no hay limitaciones en la capacidad de los inventarios.

Los autores muestran que una política de retornos con clasificación en diferentes categorías de calidad, mejora las ganancias en un 4% con respecto a una política sin clasificación, en una instalación sin límites de capacidad.

La diferencia más notoria entre los trabajos [8] y [9] es la forma en la que los retornos son clasificados. En [8] la categoría de cada retorno es conocida, mientras que en [9], se utiliza una función de densidad para determinar la probabilidad de que un retorno sea de una categoría q determinada.

Denizel et al. (2007) [6] se enfoca principalmente en la incertidumbre a la hora de clasificar los retornos que llegan en cada período, la cual se modela de forma estocástica. Si bien la cantidad de retornos que llega en un período es conocida, la calidad de cada uno de ellos no lo es. Formulan un programa estocástico para un entorno multi-período, en el que los niveles de calidad de los retornos pueden tomar cualquier configuración (de malo a bueno) basados en probabilidades preestablecidas. La programación estocástica es una metodología atractiva para éste problema porque garantiza una solución factible en todas las variantes posibles de nivel de calidad.

En este trabajo se considera un modelo con capacidad limitada, revisión periódica y un solo producto. Además, se manejan costos de remanufacturación crecientes a calidades decrecientes, y se permite la disposición final y el rescate (venta). Los retornos que no se remanufacturan ni se desechan pueden ser vendidos, el valor de rescate es más alto para los retornos de calidades más altas.

La demanda es determinística y no estacionaria, mientras que el horizonte de planificación es finito, y el backlogging está permitido.

Los retornos que son remanufacturados no presentan diferencias para el consumidor final, independientemente de su calidad al retornar. Como restricción adicional, no se pueden remanufacturar más retornos que los disponibles en cada período.

También se asume que existe uno o varios inventarios para retornos no clasificados, uno o varios inventarios para retornos ya clasificados, y uno o varios inventarios para retornos ya remanufacturados. Esto lleva a que el modelo se manejen variables tales como cuántos retornos se clasifican, cuántos retornos se remanufacturan, cuántos retornos son rescatados, en cada período. El costo de inventario de un producto remanufacturado es mayor que el costo de inventario de un retorno no remanufacturado.

El problema se formula con un enfoque de escenarios. Para cada período, se tienen dos o más entradas posibles, que se determinan mediante una variable aleatoria. Cada entrada determina un porcentaje de los retornos de calidad *q* y a cada uno de estos casos se les llama escenarios. Dado que cada escenario se vuelve a ramificar de igual manera en el siguiente período, vamos a tener 2^i ramificaciones en el período i.

La Figura 6 muestra gráficamente, un ejemplo con tres períodos.

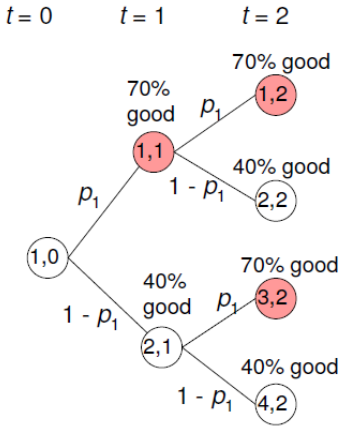


Figura 6: Construcción de escenarios.

A continuación se presenta el modelo matemático. El objetivo del mismo es maximizar las ganancias a lo largo del proceso. Se asume que los inventarios empiezan vacíos.

Los componentes del modelo:

Índices:

i: Categoría de calidades, *i*=1 (mejor), . . ., *I* (peor)
t: Período de tiempo, *t*=1, . . ., *T*
j: Escenario *j* del período *t*, *j* = 1, ..., *J(t); J(0)* = 1

Parámetros:

D_t: Demanda de productos remanufacturados en el período *t*
B_t: Cantidad de retornos que llegan al comienzo del período *t*
C_t: Capacidad permitida de remanufacturación en el período *t*
a_i: Cantidad de recursos de remanufacturación usados sobre un retorno de calidad *i*
p_r: Precio de un producto remanufacturado
c_{it}: Costo unitario de remanufacturación para un retorno de calidad *i* en el período *t*
s_{it}: Valor de rescate unitario de un retorno de calidad *i* en el período *t*

h_{it}: Costo de inventario unitario de un retorno de calidad *i* en el período *t*
h_r: Costo unitario de inventario de un producto remanufacturado
h: Costo unitario de inventario de un retorno no clasificado
π: Costo unitario por backlogging por período
g: Costo unitario por clasificar
p_t^j: Probabilidad del escenario *j* del período *t*
r_{it}^j: Fracción de elementos de calidad *i* bajo el escenario *j* del período *t*

Variables de decisión:
x_t^j: Cantidad de retornos clasificados bajo el escenario *j* del período *t*
z_{it}^j: Cantidad de retornos de calidad i remanufacturados en el período *t* bajo el escenario *j*
v_{it}^j: Cantidad de retornos de calidad *i* vendidos (rescatados) al final del período *t* y bajo el escenario *j*

Variables auxiliares:
u_{it}^j: Inventario de retornos de calidad *i* al final del período *t* bajo el escenario *j*
y_t^{j+}: Inventario de productos remanufacturados al final del período *t* bajo el escenario *j*
y_t^{j-}: Backlog de productos remanufacturados al final del período *t* bajo el escenario *j*
b_t^j: Cantidad de retornos no clasificados al final del período *t* bajo el escenario *j*

El modelo matemático propuesto es el siguiente:

$$max_{x_t^j, z_{it}^j, v_{it}^j} \Pi = \sum_{t=1}^{T} \sum_{j=1}^{J(t)} p_t^j \left\{ \sum_{i=1}^{I} \begin{pmatrix} (p_r - c_{it}) z_t^j + \\ s_{it} v_{it}^j - h_{it} u_{it}^j \end{pmatrix} - h_r y_t^{j+} - \pi y_t^{j-} \right\} - \sum_{t=1}^{T} \sum_{j=1}^{J(t-1)} p_t^j \left(g x_t^j - h b_t^j \right) \quad (76)$$

s.a:

$$\left(y_{t-1}^{pred(j,t)+} - y_{t-1}^{pred(j,t)-} \right) - \left(y_t^{j+} - y_t^{j-} \right) + \sum_{i=1}^{I} z_{it}^j = D_t \quad \forall\, t; j = 1, \dots, J(t) \quad (77)$$

$$b_t^j + x_t^j - b_{t-1}^{pred(j,t)} = B_t \quad \forall\, t; j = 1, \dots, J(t-1) \quad (78)$$

$$z_{it}^j + u_{it}^j - u_{i,t-1}^{pred(j,t)} + v_{it}^j = r_{it}^j x_t^{pred(j,t)} \quad \forall\, i; \forall\, t; j = 1, \dots, J(t) \quad (79)$$

$$\sum_{i=1}^{I} a_i z_{it}^j \leq C_t \quad \forall\, t; j = 1, \dots, J(t) \quad (80)$$

$$y_T^j = 0 \quad j = 1, \dots, J(T) \quad (81)$$

$$y_t^{j+}, y_t^{j-}, z_{it}^j, v_{it}^j, u_{it}^j, x_t^j \geq 0 \quad \forall\, i, t, j \quad (82)$$

Donde *pred(j,t)* es el escenario predecesor al escenario *(j,t)*. Por ejemplo, en la Figura 6, podemos ver que *pred(1,2)* y *pred(2,2)* son escenarios predecesores del escenario *(1,1)*, mientras que *pred(3,2)* y *pred(4,2)* de *(2,1)*, etc.

De este trabajo, se concluye que la ganancia obtenida depende de la forma que tiene la curva de costos ya que la misma está relacionada con la calidad de

los retornos, con el valor de rescate de un retorno no remanufacturado, y con el costo por clasificar. Por este motivo, es muy importante ser preciso a la hora de estimar dichos valores, ya que el resultado de la heurística depende directamente de estas estimaciones. Por otro lado, el costo de inventario y el costo unitario de backlogging tienen una incidencia menor en la solución óptima.

En Nenes et al. (2010) [21] se investigan diferentes políticas de control de inventario, para un sistema en dónde tanto la demanda de nuevos productos como los retornos (cantidad y calidad) son estocásticos. Se asumen variables independientes y randómicas.

Se estudia el caso en que las empresas, con el objetivo de satisfacer la demanda, toman las siguientes decisiones en un horizonte de planeación infinito: producción de nuevos productos, inspección de artículos retornados y remanufacturación de retornos que pasaron la inspección.

Manejan tres tipos de inventario: productos listos, retornos remanufacturables y retornos aún no inspeccionados. El costo de almacenamiento no es el mismo para cada uno de estos inventarios.

El trabajo describe la política utilizada actualmente por estas empresas e intenta determinar si es óptima (en términos de costos). También introduce nuevas políticas con diversas reglas de prioridad y criterios de decisión, según los escenarios planteados en cada caso.

Concluyen que las políticas actuales (como mantener un stock de seguridad para cubrir fluctuaciones de la demanda) incurren en un aumento del costo, en comparación con políticas alternativas y más eficaces (como probabilidad de desabastecimiento).

En Nenes & Nikolaidis (2012) [20] se propone una formulación MILP (Mixed Integer Linear Programing) con el objetivo de optimizar las decisiones de obtención de retornos, remanufacturación, almacenamiento y la recuperación (venta) de artículos retornados.

En éste trabajo se extiende el modelo presentado en Nikolaidis (2009) [22] y desarrollan un modelo multi-período, dado que un modelo de un solo período muchas veces resulta inadecuado. Para abordar un problema multi-período se podría aplicar un modelo de un solo período varias veces, pero esto no llevaría a la optimización global de los beneficios de la empresa.

El modelo es lo suficientemente flexible como para incorporar también múltiples proveedores y varios niveles de calidad de los productos devueltos.

No se maneja disposición final dado que todos los retornos pueden ser remanufacturados sin importar su nivel de calidad.

La demanda, que es cubierta sólo con productos remanufacturados, es determinística y debe ser satisfecha en cada período, no hay backlogging. Además, no se remanufacturan retornos hasta que no haya una demanda específica por productos remanufacturados.

En Zikopoulos (2012) [37] se estudia otro modelo correspondiente al problema de inventario con clasificación de retornos.

En este trabajo se tiene en cuenta el costo por escasez (faltantes), la demanda es determinística, estacionaria y conocida, y el horizonte de

planificación es infinito. La cantidad de retornos también es estacionaria, y se consideran dos niveles de calidad, buena y mala, teniéndose en cuenta la incertidumbre en los tiempos de entrega de los retornos (tiempos estocásticos), lo cual dificulta la planificación eficiente de remanufacturación y gestión de inventario. La proporción de retornos de calidad buena es una variable aleatoria con densidad de probabilidad conocida y funciones de probabilidad acumulativas. Un retorno de buena calidad requerirá menos tiempo de remanufactura que uno de mala calidad.

Una descripción del proceso es la presentada en la Figura 7:

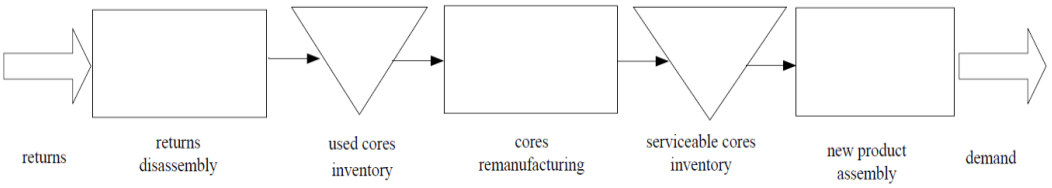


Figura 7: Proceso de remanufactura y montaje

Se puede observar que los retornos pasan por un proceso de desmontaje, donde se evalúan las piezas individualmente para luego pasar a formar parte de un inventario de retornos (artículos usados).

Cuando la demanda lo requiera, estos retornos ya clasificados se remanufacturarán para pasar a un inventario de artículos disponibles. Estos artículos disponibles se usarán para ser ensamblados y crear así, un nuevo producto que saldrá al mercado satisfaciendo la demanda.

Se busca evaluar la ganancia en ahorros, al tener en cuenta la variabilidad en la calidad de los retornos durante la planificación de la remanufactura.

Para esto se analizan dos modelos:

- Modelo A: Con una consideración explícita sobre la incertidumbre de la calidad de los retornos. Se asume que la taza de los retornos es una variable aleatoria. Si la proporción de retornos con calidad buena es *q* la proporción de retornos con calidad pobre o mala será de *1-q*. Los tiempos de remanufacturación de retornos, que difieren según su calidad, son variables también aleatorias.
- Modelo B: Ignorando la incertidumbre sobre la calidad de los retornos. Bajo este caso, hay tres sub modelos:
  - ✓ B.1: Se asume que todos los tiempos de remanufacturación son para el peor caso, de calidad mala.
  - ✓ B.2: Se asume el tiempo de remanufacturación es igual a un valor esperado.
  - ✓ B.3: Se asume que se esperan iguales proporciones de retornos con calidad buena y mala. Por lo tanto los tiempos de remanufacturación se asumen: la mitad con duración según calidad buena, y la otra mitad según calidad mala.

En este trabajo concluyen que conocer con anterioridad la variabilidad y proporción de la calidad de los retornos en cada período, o suponer distintos tiempos de remanufacturación sobre los retornos heterogéneos, puede no

resultar significativo cuando se implementa un proceso simple de planificación y políticas de manejo de inventario.

El motivo de esto es que para elegir la política más eficiente, se necesita procesar toda la información de la realidad y esto conlleva a un costo extra, que puede superar el beneficio de usar dicha información. Por lo tanto, muchas veces es mejor omitir el procesamiento de todos los datos disponibles, haciendo que la política usada sea más simple.

Los estudios numéricos realizados, bajo los supuestos de este modelo, muestran que la información sobre la tasa de retornos por calidad es útil cuando la variabilidad en la calidad es pequeña, ya que a mayor diferencia en las calidades, mayor es el costo que se requiere en inventarios. Cuanto más variabilidad entre las calidades, mayor número de inventarios se requieren, tanto así como el costo de mantenerlos seguros.

En Mahaparta et al. (2012) [18] se trabaja sobre un modelo para múltiples artículos y múltiples períodos, donde la demanda es satisfecha con el resultado de la manufacturación y remanufacturación. Los retornos se consideran heterogéneos, por tanto se los clasifica, y estos pueden ser remanufacturados o desechados. El nivel de calidad que se le asigna a un retorno, está relacionado con la cantidad de horas de trabajo que requiere para llevarlo a un estado "como nuevo".

Se contemplan costos unitarios de adquisición (esto abarca recolección, empaquetado, envío y ordenamiento de los retornos para remanufacturar), costos unitarios de remanufacturación, costos unitarios de producción variables, costos de inventario y costos fijos de producción. Las funciones de costos son no lineales. Cuenta con restricciones de capacidad de producción. Los artículos listos, manufacturados o remanufacturados ya que no hay distinción en la calidad, contienen un número de versión. La tasa de retornos así como la de adquisición de insumos se considera no uniforme.

Proponen un enfoque de programación lineal entera mixto (MILP) para el problema.

Los resultados del análisis numérico que realizan, indican que el plan de producción óptimo depende de la calidad de los retornos, de la segregación de estos, y de la capacidad de reajuste de los costos.

5. Conclusiones

A modo de conclusión se puede decir que en este documento hemos intentado brindar información sobre los diferentes problemas de inventario, modelos y algoritmos que se han estudiado en los últimos años, poniéndose el foco en aquellos problemas de control de inventario que tienen características similares al problema estudiado en el marco del proyecto de grado, y que presentan remanufacturación y clasificación de los retornos.

Durante las últimas décadas el interés por incrementar ganancias y reducir costos a nivel de la industria, ha hecho que varios expertos estudien diferentes maneras de optimizar procesos de producción. Esto ha derivado en estudios y trabajos a lo largo de los años, exponiendo una gran cantidad de variantes con respecto a la manufacturación y remanufacturación de artículos. Cada uno de estos estudios y trabajos son basados en trabajos anteriores, extendiéndolos, agregando variaciones a los modelos existentes, o analizando resultados desde perspectivas y contextos diferentes.

En la literatura consultada se han estudiado problemas de inventario con diferentes características, como por ejemplo modelos donde la demanda es satisfecha solo con remanufacturación de retornos, otros donde la producción también es considerada. Modelos más recientes comenzaron a introducir la clasificación de los retornos según su calidad en no más de dos o tres niveles, afectando los costos y tiempos de remanufacturación. Otras características son: uno o múltiples artículos dependientes entre sí, múltiples períodos, costos fijos o variables, funciones de costos lineales o cóncavas, considerando en algunos casos costos de configuración. En algunos casos se considera una incertidumbre en los tiempos de entrega de los retornos. También la calidad de los retornos se asume conocida o no en los diferentes modelos.

En varios trabajos se amplía el modelo clásico presentado por Wagner-Whitin y se generaliza su algoritmo. Se estudia y compara el desempeño computacional de versiones modificadas de las heurísticas Silver Meal (SM), Least Unit Cost (LUC) y Part Period Balancing (PPB) para sistemas con manufacturación y remanufacturación, obteniendo mejoras en los resultados.

Para la resolución del problema se han propuesto diferentes técnicas como: un algoritmo de programación dinámica, un procedimiento Basic Tabu Search (BTS), una metaheurística Variable Neighborhood Search (VNS), el algoritmo Differential Evolution (DE). En algunos estudios se propone un modelo matemático MILP (Mixed Integer Linear Programing) o se investigan diferentes políticas de control de inventario.

En general se concluye que los costos totales descienden si se tiene en cuenta la clasificación de los retornos heterogéneos, frente a los modelos donde no existe dicha clasificación.

6. Referencias

[1] Aggarwal A, Park J, 1993. Improved algorithms for economic lot-size problems. *Operations Research*, 14(1), pp. 549–571.

[2] Atamtürk A, Küçükyavuz S, 2008. An $O(T^2)$ algorithm for the lost sizing with inventory bounds and fixed costs. *Operations Reserch Letters,* 36(1), pp. 297–299.

[3] Baki M, Chaouch B, Abdul-Kader W, 2014. A heuristic solution procedure for the dynamic lot sizing problem with remanufacturing and product recovery. *Computers & Operations Research*, 43(1), pp. 225–236.

[4] Brahimi N, Dauzere-Peres S, Najid N, Nordli A, 2006. Single item lot sizing problems. *European Journal of Operational Research*, 168(1), pp. 1–16.

[5] De Brito M, Dekker R, 2002. Reverse Logistics – a framework. *Erasmus University Rotterdam, Econometric Institute Report* EI 2002-38.

[6] Denizel M, Ferguson M, Souza G, 2007. Multi-Period Remanufacturing Planning With Uncertain Quality of Inputs. *Engineering Management*, 57(3), pp. 394 – 404.

[7] Federgruen A, Tzur M, 1991. A simple forward algorithm to solve general dynamic lot sizing models with n periods in *O(n log n)* or *O(n)* time. *Management Science*, 37(1), pp. 909–925.

[8] Ferguson M, Guide D, Koca E, Souza G, 2006. Remanufacturing Planning with Different Quality Levels for Product returns. *Robert H. Smith School Research* Paper No. RHS 06-050.

[9] Ferguson M, Guide D, Koca E, Souza G, 2009. The Value of Quality Grading in Remanufacturing. *Production and Operations Management*, 18(3), pp. 300–314.

[10] Ferrer E, Mouriz A, Piñeyro P, 2002. Políticas de inventarios, mantenimiento y pronósticos. Anexo B Control de Inventario. Taller V, *Departamento de Investigación Operativa, Instituto de Computación, Facultad de Ingeniería*.

[11] Ford H, 1913. How Many Parts to Make at Once. Reprinted from *Factory, The Magazine of Management,* 10(2), pp. 135-136, 152.

[12] Golany B, Yang J, Yu G, 2001. Economic lot-sizing with remanufacturing options. *IIE Transactions*, 33(1), pp. 995-1003.

[13] Guide D, 2000. Production planning and control for remanufacturing: industry practice and research needs. *Journal of Operations Management,* 18(1), pp. 467–483.

[14] Guner-Goren H, Tunali S, Jans R, 2010. A review of applications of genetic algorithms in lot sizing. *Journal of Intelligent Manufacturing*, 21(1), pp. 575–590.

[15] Gutowski T, Sahni S, Boustani A, Graves S, 2011. Remanufacturing and Energy Savings. *Environmental Science & Technology*, 45(1), pp. 4540–4547.

[16] Ijomah W, 2002. A model-based definition of the generic remanufacturing business process. *Plymouth Business School*.

[17] Li X, Baki F, Tian P, Chaouch B, 2014. A robust block-chain based tabu search algorithm for the dynamic lot sizing problem with product returns and remanufacturing. *Omega*, 42(1), pp. 75–87.

[18] Mahapatra S, Pal R, Narasimahan R, 2012. Hybrid (re)manufacturing: manufacturing and operational implications. *International Journal of Production Research*, 50(14), pp. 3786-3808.

[19] Modelo EOQ, https://en.wikipedia.org/wiki/Economic_order_quantity

[20] Nenes G, Nikolaidis Y, 2012. A Multi–period Model for Managing Used Product Returns. *International Journal of Production Research*, 50(5), pp. 1360-1376.

[21] Nenes G, Panagiotidou S, Dekker R, 2010. Inventory control policies for inspection and remanufacturing of returns: A case study. *Int. J. Production Economics*, 125(1), pp. 300–312.

[22] Nikolaidis Y, 2009. A modelling framework for the acquisition and remanufacturing of used products. *International Journal of Sustainable Engineering*, 2(3), pp. 154–170.

[23] Parsopoulos K, Konstantaras I, Skouri K, 2015. Metaheuristic optimization for the Single-Item Dynamic Lot Sizing problem with returns and remanufacturing. *Computers & Industrial Engineering*, 83(1), pp. 307–315.

[24] Piñeyro P, 2013. An analysis of the economic lot-sizing problem with return options focused on the remanufacturing plan. *Tesis de Doctorado en Informática, PEDECIBA*.

[25] Piñeyro P, Viera O, 2009. Inventory policies for the economic lot-sizing problem with remanufacturing and final disposal options. *Journal of Industrial and Management Optimization,* 5(2), pp. 217–238

[26] Piñeyro P, Viera O, 2015. An improved Tabu Search based on procedure for the economic lot-sizing problem with remanufacturing. *Proceeding of ICoR 2015, Ámsterdam, Holanda*.

[27] Richter K, Sombrutzki M, 2000. Remanufacturing planning for the reverse Wagner/Whitin models. *European Journal of Operational Research*, 121(1), pp. 304-315.

[28] Richter K, Weber J, 2001. The reverse Wagner/Whitin model with variable manufacturing and remanufacturing cost. *International Journal of Production Economics*, 71(1), pp. 447-456.

[29] Schulz T, 2011. A new Silver–Meal based heuristic for the single-item dynamic lot sizing problem with returns and remanufacturing. *International Journal of Production Research*, 49(9), pp. 2519-2533.

[30] Sifaleras A, Konstantaras I, Mladenović N, 2015. Variable neighborhood search for the economic lot sizing problem with product returns and recovery. *Int. J. Production Economics*, 160(1), pp. 133–143.

[31] Silver E, Pyke D, Peterson R, 1998. Inventory Management and Production Planning and Scheduling. *3rd ed. New York: Wiley*.

[32] Teunter R, Bayindir Z, Van Den Heuvel W, 2006. Dynamic lot sizing with product returns and remanufacturing. *International Journal of Production Research*, 44(20), pp. 4377-4400.

[33] Wagelmans A, Van Hoesel C, Kolen A, 1992. Economic lot sizing: An O(nlog n) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research*, 40(1), pp. 145–156.

[34] Wagner H, Whitin T, 1958. Dynamic Version of the Economic Lot Size Model**.** *Management Science*, 5(1), pp. 89–96.

[35] Yang J, Golany B, Yu G, 2005. A Concave-Cost Production Planning Problem with Remanufacturing Options. *Wiley Periodicals, Inc. Naval Research Logistics*, 52(1), pp. 443–458.

[36] Zangwill W, 1968. Minimum concave cost flows in certain networks. *Management Science*, 14(7), pp. 429–450.

[37] Zikopoulos C, 2012. Remanufacturing lot-sizing under alternative perceptions of returned units quality. *Int. Journal of Business Science and Applied Management*, 7(3).

Paquete GNU Linear Programming Kit (GLPK)

Anexo 2

Proyecto de grado

Luciano Alvarez
Gabriela Arriola
Matilde Macciò

Supervisor: Pedro Piñeyro

Tabla de contenido

1. Modelo matemático

La definición del modelo matemático consiste en un conjunto de parámetros, variables, una función objetivo y sus restricciones. En GLPK, este modelo debe definirse en un archivo *.mod* y es pasado como parte de los parámetros para la ejecución. En la Sección 4 se encuentra el archivo del modelo ELSRC, y se describe a continuación.

1.1 Parámetros

Los parámetros del modelo son los datos de entrada que recibe GLPK y que servirán para determinar la solución del problema. Los mismos se describen en la Tabla 1:

Parámetro	Descripción
T	horizonte de planificación (cantidad de períodos)
I	cantidad de niveles de calidad para los retornos
D	vector con cantidad demandada por período
U	matriz con cantidad de productos que retornan por calidad y período
Q	vector con cantidad de retornos no recuperables por período
s	matriz con valor residual unitario de venta, de un retorno por calidad y período
Kp	vector con costo de preparación para producción por período
Kr	matriz con costo de preparación para remanufacturación de retornos por calidad y período
Kd	vector con costo de preparación para disposición final de retornos por período
cp	vector con costo unitario de producir por período
cr	matriz con costo unitario de remanufacturar un retorno por calidad y período
cd	vector con costo unitario de desechar un retorno por período
hu	matriz con costo unitario de mantener en inventario un retorno por calidad y período
hq	vector con costo unitario de mantener en inventario un retorno no recuperable por período
hs	vector con costo unitario de mantener en inventario un producto listo por período
M	valor máximo entre los valores acumulados entre 1 y T para: la demanda, los retornos por calidad y los retornos no recuperables.

Tabla 1: Parámetros del modelo

Para la ejecución del algoritmo genético y de GLPK se utilizó el mismo archivo de entrada (con los parámetros antes mencionados), con el objetivo de

comparar posteriormente los resultados obtenidos. En la Sección 5 se presenta un ejemplo del archivo de entrada.

1.2 Variables de decisión

Las variables de decisión del modelo determinan la solución del problema. En la Tabla 2 se detallan las mismas para el problema estudiado:

Variable	Descripción
p	vector con cantidad a producir por período
r	matriz con cantidad de retornos de calidad i a remanufacturar en el período t
d	vector con cantidad de retornos para disposición final que se desechan en el período t
v	matriz con cantidad de retornos de calidad i que son vendidos en el período t
yr	matriz con inventario de productos a remanufacturar de calidad i en el período t
yq	vector con inventario de productos a desechar en el período t
ys	vector con inventario de productos remanufacturados y nuevos (listos) en el período t
zp	vector que indica en cada período si se realiza la actividad de producción (1 se produce, 0 no)
zr	matriz que indica en por cada período y nivel de calidad de los retornos, si se realiza la actividad de remanufacturación (1 se remanufactura, 0 no)
zd	vector que indica en cada período si se realiza la actividad de disponer finalmente (1 se desecha, 0 no)

Tabla 2: Variables de decisión del modelo

1.3 Función objetivo

La función objetivo define el problema que se desea optimizar.

En el ELSRC se desea obtener el costo mínimo de producción y remanufacturación de retornos para satisfacer la demanda en un período de tiempo, considerando además venta y disposición de retornos.

1.4 Restricciones

Las restricciones son las condiciones que debe cumplir la solución y fueron definidas en el modelo matemático.

En el presente problema existen tres tipos de restricciones:

- ✓ restricciones de inventario: son las que determinan los valores iniciales de los inventarios. En este caso, todos los inventarios comienzan vacíos.

- ✓ restricciones de equilibrio: son las que mantienen la relación entre la cantidad de elementos en los inventarios, cantidad de ítems a producir y remanufacturar, ítems demandados, retornos, etc., en cada período.
- ✓ restricciones de máximos: se define un máximo M que acota las cantidades a producir, remanufacturar y desechar, con el fin de restringir el universo del problema.

2. Archivo de entrada

El archivo de entrada utilizado para ejecutar el algoritmo genético y GLPK contiene los diferentes parámetros que se describieron en la Sección 1.1.

En la Sección 5 se presenta un ejemplo del mismo.

Primero se especifican la cantidad de períodos (T) y la cantidad de niveles de calidad (I) del problema.

Luego se define la cantidad demandada (D) para cada uno de los T períodos. Dado que existe una demanda diferente para cada período, estos valores se representan con dos columnas de largo T. La primera columna representa el período y la segunda columna representa la demanda en el correspondiente período. Por ejemplo, si tenemos que T = 3, un ejemplo de demanda sería:

```
param D:=
        1 102
        2 110
        3 107;
```

El ejemplo anterior indica que se demandaron 102, 110 y 107 elementos en los períodos 1, 2 y 3 respectivamente.

El siguiente parámetro es la cantidad de retornos que llegan (U) de calidad *i* en el período *t*. Dado que existe una cantidad de retornos diferente para cada calidad en cada período, estos valores se representan con tres columnas de largo T*I. La primera columna representa el nivel de calidad del retorno, la segunda columna representa el período, y la tercera columna representa la cantidad de retornos. Por ejemplo, si tenemos que T = 3 e I = 2, un ejemplo de cantidad de retornos sería:

```
param U:=
        1 1 25
        1 2 26
        2 1 24
        2 2 22
        3 1 21
        3 2 27;
```

El ejemplo anterior indica que se hubieron 25 retornos de calidad 1 en el período 1, 26 retornos de calidad 1 en el período 2, 24 retornos de calidad 2 en el período 1, y así sucesivamente.

Para el resto de los parámetros:

- Los que dependen de la calidad y el período, se representan en 3 columnas de largo T*I, donde la primera representa el nivel de calidad del retorno, la segunda representa el período, y la tercera el valor del parámetro que se está definiendo.

- Los que dependen solo del período, se representan en 2 columnas de largo T, donde la primera representa el período, y la segunda el valor del parámetro que se está definiendo.

Luego se define la constante M, la cual se describió anteriormente.

3. Ejemplo de Ejecución

Para ejecutar GLPK desde consola se utilizó el siguiente comando:

glpsol --tmlim 1800 -m ELSRC.mod -d entrada.dat -o salida.sol

dónde:

- *glpsol* es el nombre del comando para ejecutar GLPK.

- *--tmlim* indica el parámetro con la cantidad de segundos máximo que debe correr GLPK. Si se llega a dicho tiempo y no se ha encontrado una solución factible, GLPK cortará la ejecución devolviendo la mejor solución encontrada hasta ese momento. En este ejemplo, la ejecución se cortará a los 1800 segundos (30 minutos).

- *-m* indica el parámetro del archivo que contiene el modelo a ejecutar. Como se mencionó en la Sección 1, el modelo está contenido en el archivo *ELSRC.mod* el cual se presenta en la Sección 4.

- *-d* indica el parámetro del archivo que contiene los datos de entrada a GLPK. Un ejemplo del mismo se presenta en la Sección 5.

- *-o* indica el parámetro del archivo de salida que retorna GLPK.

4. Archivo ELSRC.mod

```
# Parámetros
param T > 0;                        #T: horizonte de planeación.
param I > 0;                        #I: cantidad de clasificaciones.
param D{t in 1..T} >= 0;            #D: demanda en el período t.
param U{i in 1..I, t in 1..T} >= 0;  #U: cantidad de productos de calidad i que retornan en el período t
param Q{t in 1..T} >= 0;            #Q: cantidad de retornos no recuperables en el período t.
param s{i in 1..I, t in 1..T};      #s: valor residual unitario de venta de un retorno de calidad i en el período t.
param Kp{t in 1..T} >= 0;           #Kp: costo de preparación para producción en el período t.
param Kr{i in 1..I, t in 1..T} >= 0;  #Kr: costo de preparación para remanufacturación de retornos de calidad i en el período t.
param Kd{t in 1..T} >= 0;           #Kd: costo de preparación para disposición final de retornos en el período t.
param cp{t in 1..T} >= 0;           #cp: costo unitario de producir en el período t.
param cr{i in 1..I, t in 1..T} >= 0;  #cr: costo unitario de remanufacturar un retorno de calidad i en el período t.
param cd{t in 1..T} >= 0;           #cd: costo unitario de disposición final de un retorno en el período t.
param hu{i in 1..I, t in 1..T} >= 0;  #hu: costo unitario de mantener en inventario un retorno de calidad i en el período t.
param hq{t in 1..T} >= 0;           #hq: costo unitario de mantener en inventario un retorno no recuperable en el período t.
param hs{t in 1..T} >= 0;           #hs: costo unitario de mantener en inventario un producto remanufacturado o nuevo en el
                                        período t.
param M >= 0;

# Variables de decisión
var p{t in 1..T} >= 0;                  #p: cantidad de productos nuevos a producir en el período t
var r{i in 1..I, t in 1..T} >= 0;       #r: cantidad de retornos de calidad i a remanufacturar en el período t
var d{t in 1..T} >= 0;                  #d: cantidad de retornos para disposición final que se desechan en el
                                        período t
var v{i in 1..I, t in 1..T} >= 0;       #v: cantidad de retornos de calidad i que son vendidos en el período t
var yr{i in 1..I, t in 0..T} >= 0;      #yr: inventario de productos a remanufacturar de calidad i en el
                                        período t
var yq{t in 0..T} >= 0;                 #yq: inventario de productos a desechar en el período t
var ys{t in 0..T} >= 0;                 #ys: inventario de productos remanufacturados y nuevos (listos) en el
                                        período t
var zp{t in 1..T}, binary;              #δp: 1 si en el período t la cantidad a producir  es positiva, 0 en caso
                                        contrario
var zr{i in 1..I, t in 1..T}, binary;   #δr: 1 si en el período t la cantidad a remanufacturar de retornos de
                                        calidad i es positiva, 0 en caso contrario
var zd{t in 1..T}, binary;              #δd: 1 si en el período t la cantidad a disponer finalmente es positiva, 0
                                        en caso contrario
# Función Objetivo
minimize ELSRC: sum{t in 1..T}(sum{i in 1..I}( Kr[i,t]*zr[i,t] + cr[i,t]*r[i,t] - s[i,t]*v[i,t] + hu[i,t]*yr[i,t]) + Kp[t]*zp[t] +
cp[t]*p[t] + Kd[t]*zd[t] + cd[t]*d[t] + hq[t]*yq[t] + hs[t]*ys[t]);

# Restricciones
s.t. inventario_remanufacturar_inicial{i in 1..I}: yr[i,0]=0;
s.t. inventario_desechar_inicial: yq[0]=0;
s.t. inventario_listos_inicial: ys[0]=0;
s.t. ec_equilibrio_inv_listos{t in 1..T}: ys[t] = ys[t-1] + p[t] + sum{i in 1..I}(r[i,t]) - D[t];
s.t. ec_equilibrio_inv_rem{i in 1..I, t in 1..T}: yr[i,t] = yr[i,t-1] + U[i,t] - r[i,t] - v[i,t];
s.t. ec_equilibrio_inv_desechar{t in 1..T}: yq[t] = yq[t-1] - d[t] + Q[t];
s.t. maximo_acumulados: M >= max(sum{t in 1..T}(D[t]), sum{i in 1..I}(sum{t in 1..T}(U[i,t])), sum{t in 1..T}(Q[t]));
s.t. maximo_delta_producir{t in 1..T}: M*zp[t] >= p[t];
s.t. maximo_delta_remanufacturar{i in 1..I, t in 1..T}: M*zr[i,t] >= r[i,t];
s.t. maximo_delta_desechar{t in 1..T}: M*zd[t] >= d[t];

end;
```

5. Ejemplo de archivo de entrada *.dat*

```
data;
param T:= 12;
param I:= 3;
param D:=
        1 102
        2 110
        3 107
        4 102
        5 99
        6 98
        7 76
        8 111
        9 115
        10 85
        11 103
        12 88;
param U:=
        1 1 25
        1 2 26
        1 3 24
        1 4 26
        1 5 24
        1 6 24
        1 7 25
        1 8 26
        1 9 25
        1 10 25
        1 11 25
        1 12 25
        2 1 24
        2 2 25
        2 3 26
        2 4 25
        2 5 24
        2 6 25
        2 7 25
        2 8 26
        2 9 26
        2 10 25
        2 11 25
        2 12 26
        3 1 24
        3 2 24
        3 3 25
        3 4 25
        3 5 24
        3 6 24
        3 7 24
        3 8 26
        3 9 28
        3 10 25
        3 11 26
        3 12 24;
param s:=
        1 1 9
        1 2 10
        1 3 10
        1 4 9
        1 5 9
        1 6 9
        1 7 8
        1 8 10
        1 9 10
        1 10 8
        1 11 9
        1 12 8
        2 1 7
        2 2 8
        2 3 8
        2 4 7
```

```
            2 5 7
            2 6 7
            2 7 6
            2 8 8
            2 9 8
            2 10 6
            2 11 7
            2 12 6
            3 1 5
            3 2 6
            3 3 6
            3 4 5
            3 5 5
            3 6 5
            3 7 4
            3 8 6
            3 9 6
            3 10 4
            3 11 5
            3 12 4;
param Kp:=
            1 538
            2 544
            3 459
            4 479
            5 515
            6 490
            7 474
            8 524
            9 487
            10 512
            11 521
            12 486;
param Kr:=
            1 1 149
            1 2 56
            1 3 88
            1 4 138
            1 5 52
            1 6 84
            1 7 50
            1 8 122
            1 9 61
            1 10 84
            1 11 61
            1 12 85
            2 1 249
            2 2 156
            2 3 188
            2 4 238
            2 5 152
            2 6 184
            2 7 150
            2 8 222
            2 9 161
            2 10 184
            2 11 161
            2 12 185
            3 1 349
            3 2 256
            3 3 288
            3 4 338
            3 5 252
            3 6 284
            3 7 250
            3 8 322
            3 9 261
            3 10 284
            3 11 261
            3 12 285;
param cp:=
            1 123
            2 181
            3 106
```

```
		4 182
		5 157
		6 188
		7 161
		8 111
		9 178
		10 196
		11 101
		12 155;
param cr:=
		1 1 11
		1 2 12
		1 3 11
		1 4 20
		1 5 14
		1 6 18
		1 7 13
		1 8 10
		1 9 10
		1 10 17
		1 11 16
		1 12 15
		2 1 21
		2 2 22
		2 3 21
		2 4 30
		2 5 24
		2 6 28
		2 7 23
		2 8 20
		2 9 20
		2 10 27
		2 11 26
		2 12 25
		3 1 31
		3 2 32
		3 3 31
		3 4 40
		3 5 34
		3 6 38
		3 7 33
		3 8 30
		3 9 30
		3 10 37
		3 11 36
		3 12 35;
param hu:=
		1 1 4
		1 2 3
		1 3 4
		1 4 4
		1 5 3
		1 6 4
		1 7 3
		1 8 3
		1 9 4
		1 10 3
		1 11 3
		1 12 4
		2 1 3
		2 2 2
		2 3 3
		2 4 3
		2 5 2
		2 6 3
		2 7 2
		2 8 2
		2 9 3
		2 10 2
		2 11 2
		2 12 3
		3 1 2
		3 2 1
		3 3 2
```

```
            3 4 2
            3 5 1
            3 6 2
            3 7 1
            3 8 1
            3 9 2
            3 10 1
            3 11 1
            3 12 2;
param hs:=
            1 17
            2 16
            3 11
            4 16
            5 12
            6 16
            7 14
            8 13
            9 12
            10 15
            11 11
            12 10;
param M:=16425;
param Q:=
            1 5
            2 6
            3 4
            4 6
            5 4
            6 4
            7 5
            8 6
            9 5
            10 5
            11 5
            12 5;
param Kd:=
            1 47
            2 24
            3 47
            4 45
            5 36
            6 42
            7 27
            8 40
            9 49
            10 46
            11 43
            12 41;
param cd:=
            1 5
            2 8
            3 10
            4 10
            5 10
            6 7
            7 9
            8 6
            9 9
            10 5
            11 7
            12 9;
param hq:=
            1 1
            2 1
            3 1
            4 1
            5 2
            6 2
            7 2
            8 1
            9 1
            10 1
            11 1
```

```
        12 2;

end;
```

Algoritmo Genético

Anexo 3

Proyecto de grado

Luciano Alvarez
Gabriela Arriola
Matilde Macciò

Supervisor: Pedro Piñeyro

Tabla de contenido

1. Configuración de biblioteca ECJ

Se debe contar con la variable de entorno CLASSPATH, que contenga la ruta a la carpeta *"...\ecj"* donde se encuentre la biblioteca.

Para la ejecución del algoritmo por línea de comandos, se debe estar posicionado en la ruta *"...\ecj\ec\app\elsrc"*.

Por más información sobre la biblioteca ECJ acceder a https://cs.gmu.edu/~eclab/projects/ecj

2. Compilación del algoritmo genético

En caso de necesitar compilar el algoritmo, posicionado desde *"CLASSPATH\ecj\ec\app\elsrc"* se debe ejecutar el siguiente comando:

javac ../../Evolve.java CruzamientoPorActividad.java MutacionPorActividad.java ELSRCStatistics.java ELSRCStatisticsWebView.java

3. Ejecución del algoritmo por línea de comandos

Para ejecutar el algoritmo, desde la ruta *"CLASSPATH\ecj\ec\app\elsrc"* ejecutar el siguiente comando:

java ec.Evolve -file ELSRC.params

El archivo *ELSRC.params* contiene todos los parámetros del algoritmo. Estos pueden ser modificados en este archivo o pueden ser pasados por línea de comandos agregando por cada uno que se desee modificar el siguiente segmento:

-p nombreDeParametro=valor

Notar que no lleva espacios entre el nombre del parámetro, el símbolo de igualdad y el valor.

Un ejemplo de ejecución sería:

java ec.Evolve -file ELSRC.params -p eval.problem.input_file=inputFile.dat

donde *eval.problem.input_file* es el parámetro que indica el archivo de entrada.

4. Parámetros del algoritmo

La Tabla 1 describe los parámetros relevantes al algoritmo ELSRC.

Parámetro	Descripción	Particular de ELSRC
generations	cantidad de veces que el algoritmo crea una nueva generación	
jobs	cantidad de ejecuciones del algoritmo	
conDisposicionFinal	*true* modela el problema con disposición final de retornos *false* en caso contrario	✓
conVenta	*true* modela el problema con venta de retornos *false* en caso contrario	✓
costosUnitariosFijos	*true* modela el problema con costos unitarios fijos *false* si se quiere trabajar con costos unitarios variables	✓
pop.subpop.0.size	tamaño de la población	
pop.subpop.0.species	tipo de la representación de la especie de los individuos	
pop.subpop.0.species.ind	tipo de la representación del individuo	
pop.subpop.0.species.ind.T	cantidad de períodos del problema.	✓
pop.subpop.0.species.ind.I	cantidad de calidades de retornos del problema.	✓
pop.subpop.0.species.genome-size	tamaño del genoma que representa el individuo	
pop.subpop.0.species.crossover-type	tipo de cruzamiento	
pop.subpop.0.species.mutation-type	tipo de mutación	
pop.subpop.0.species.pipe	clase que implementa el operador de mutación a utilizar	
pop.subpop.0.species.pipe.mutation-prob	probabilidad de aplicar mutación por actividad, sobre cada alelo (posición) del genoma.	✓
pop.subpop.0.species.pipe.probAjuste	probabilidad de que se apliquen ajustes en producción en el momento de la mutación	✓
pop.subpop.0.species.pipe.source.0	clase que implementa el operador de cruzamiento a utilizar	
pop.subpop.0.species.pipe.source.0.likelihood	probabilidad de aplicar cruzamiento sobre el genoma	
pop.subpop.0.species.pipe.source.0.source.0	clase que implementa el operador de selección a utilizar sobre el primer padre que participará del cruzamiento	
pop.subpop.0.species.pipe.source.0.source.1	clase que implementa el operador de selección a utilizar sobre el segundo padre que participará del cruzamiento	
select.tournament.size	indica, para la selección por torneo, la	

	cantidad de participantes en el mismo
seed.0	semilla a utilizar para generar números aleatorios
stat	clase encargada de presentar los resultados y estadísticas de las ejecuciones
eval.problem.input_file	archivo de entrada
silent	*true* para que no se imprima ningún log de salida a pantalla *false* en caso contrario

Tabla 1. Parámetros del algoritmo.

A continuación, se muestra un ejemplo del contenido del archivo de parámetros:

```
parent.0            = ../../simple/simple.params
eval.problem        = ec.app.elsrc.ELSRC
state         = ec.simple.SimpleEvolutionState

generations         = 10
jobs = 1
quit-on-run-complete      = true

# Parametros del problema
conDisposicionFinal = true
conVenta = true
costosUnitariosFijos = false

# Defino las variables generales al problema
pop.subpop.0.size = 100

pop.subpop.0.species              = ec.vector.IntegerVectorSpecies
pop.subpop.0.species.fitness      = ec.simple.SimpleFitness

pop.subpop.0.species.ind  = ec.vector.IntegerVectorIndividual
pop.subpop.0.species.ind.T = 1
pop.subpop.0.species.ind.I = 1
pop.subpop.0.species.min-gene=0
pop.subpop.0.species.max-gene=100000

pop           = ec.Population
init          = ec.simple.SimpleInitializer
finish        = ec.simple.SimpleFinisher
breed         = ec.simple.SimpleBreeder
eval          = ec.simple.SimpleEvaluator
exch          = ec.simple.SimpleExchanger

pop.subpop.0.species.genome-size = 10
pop.subpop.0.species.crossover-type     = two
pop.subpop.0.species.mutation-type      = reset
pop.subpop.0.species.mutation-prob      = 0.1

pop.subpop.0.species.pipe               = ec.app.elsrc.MutacionPorActividad
pop.subpop.0.species.pipe.mutation-prob = 0.3
pop.subpop.0.species.pipe.probAjuste = 0.65
pop.subpop.0.species.pipe.likelihood          = 1
pop.subpop.0.species.pipe.source.0            =
ec.app.elsrc.CruzamientoPorActividad
pop.subpop.0.species.pipe.source.0.likelihood       = 0.5
pop.subpop.0.species.pipe.source.0.source.0   = ec.select.TournamentSelection
pop.subpop.0.species.pipe.source.0.source.1   = same
select.tournament.size                        = 2
```

```
seed.0       = time

# Add statistics object
stat.do-generation = false
stat.do-message = false
stat.do-description = false
#stat        = ec.app.elsrc.ELSRCStatistics
stat         = ec.app.elsrc.ELSRCStatisticsWebView

# Archivo de entrada
eval.problem.input_file = input_file.dat

silent = true
```

5. Entrada del algoritmo

El archivo de entrada *.dat* utilizado por el algoritmo genético es el mismo que el que se utiliza como entrada por GLPK. Para más información sobre dicho archivo, véase su descripción en el Anexo 2 de GLPK.

6. Salidas del algoritmo

Existen dos salidas posibles y se puede elegir entre una u otra a través del parámetro *stat*.

Si *stat=ec.app.elsrc.ELSRCStatistics* se genera un archivo de texto con extensión *.out.stat* por cada ejecución independiente del algoritmo. Este archivo contiene información de la solución al problema, como cantidades a producir, remanufacturar, vender y disponer por período, y también datos que permiten evaluar el algoritmo, tales como tiempo del algoritmo, tiempo en hallar la mejor solución, cómo fue evolucionando la solución, en qué número de generaciones se halló el mejor individuo, etc.

Este es un ejemplo de un archivo de salida generado para una ejecución:

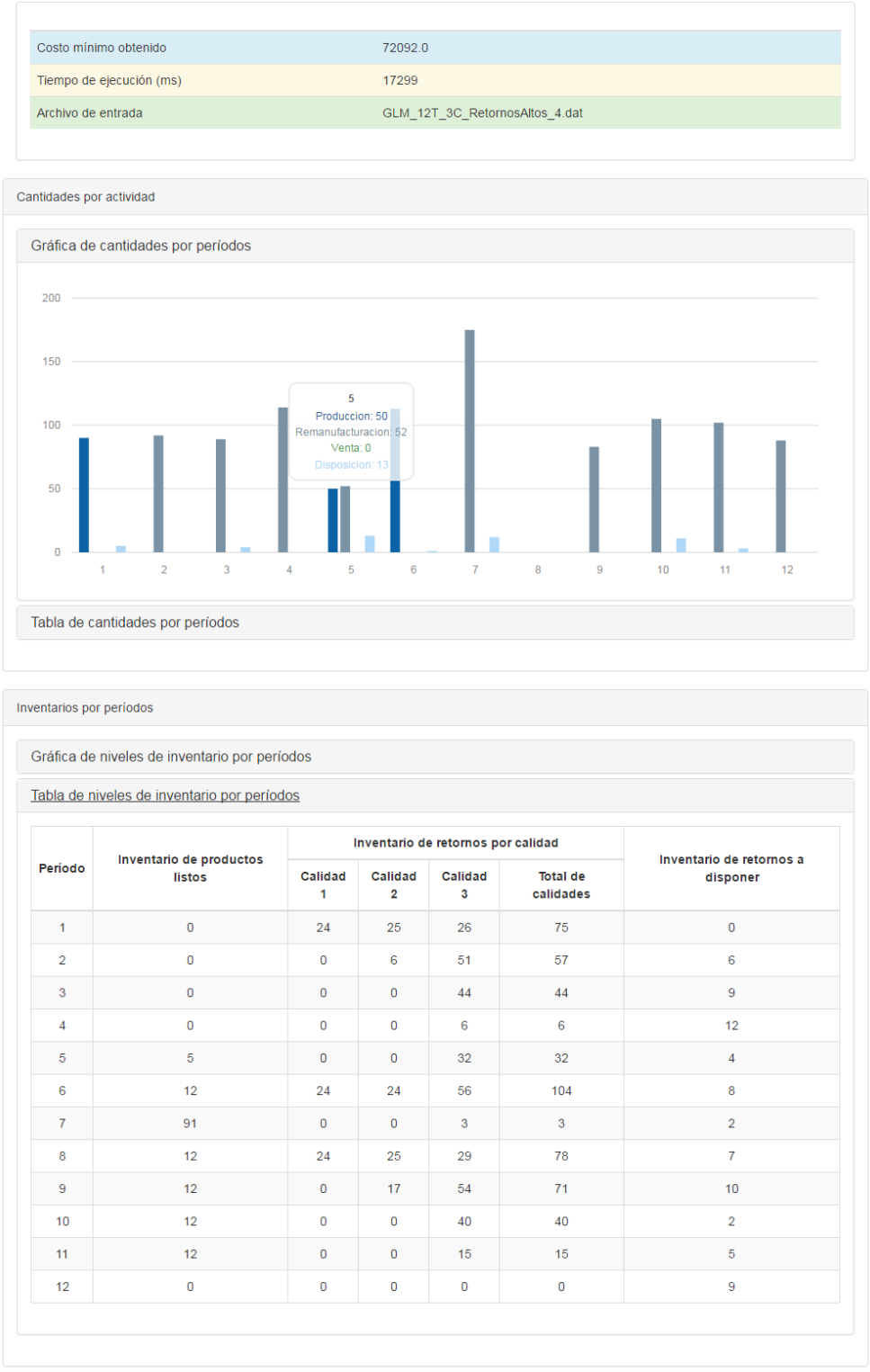
```
   Archivo de datos de entrada: GLM_12T_3C_RetornosAltos_4.dat
   Tiempo ejec.(ms): 17820
   Tiempo mejor fitness (ms): 3108
   Evolucion mejor fitness cada 100 generaciones:
   100 -71020.0
   200 -71020.0
   300 -71020.0
   400 -71020.0
   Num. generaciones del mejor fitness: 71
   Mejor fitness: -71020.0
   Genotipo: 38 0 83 0 0 44 0 75 0 0 0 18 24 24 26 26 27 24 25 0 50 24 26 25
25 25 25 25 25 24 26 0 33 42 25 25 26 25 0 50 26 0 47 0 0 39 51 30 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 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 5 0 0 3 0 0 0 16 0
10 0 13
   La solucion hallada es factible!!!
   Costo Total: 71020.0
   Solucion del ELSRC:
   No.       Variable           Valor
   ---       --------           -----
   1         p[1]               38
   2         p[2]               0
   3         p[3]               83
   4         p[4]               0
   5         p[5]               0
   6         p[6]               44
   7         p[7]               0
   8         p[8]               75
   9         p[9]               0
   10        p[10]              0
   11        p[11]              0
   12        p[12]              18
   13        r[1,1]             24
   14        r[2,1]             25
   15        r[3,1]             26
   16        r[1,2]             24
   17        r[2,2]             25
   18        r[3,2]             25
   19        r[1,3]             26
```

20	r[2,3]	25
21	r[3,3]	0
22	r[1,4]	26
23	r[2,4]	25
24	r[3,4]	50
25	r[1,5]	27
26	r[2,5]	25
27	r[3,5]	26
28	r[1,6]	24
29	r[2,6]	24
30	r[3,6]	0
31	r[1,7]	25
32	r[2,7]	26
33	r[3,7]	47
34	r[1,8]	0
35	r[2,8]	0
36	r[3,8]	0
37	r[1,9]	50
38	r[2,9]	33
39	r[3,9]	0
40	r[1,10]	24
41	r[2,10]	42
42	r[3,10]	39
43	r[1,11]	26
44	r[2,11]	25
45	r[3,11]	51
46	r[1,12]	25
47	r[2,12]	25
48	r[3,12]	30
49	d[1]	5
50	d[2]	0
51	d[3]	0
52	d[4]	3
53	d[5]	0
54	d[6]	0
55	d[7]	0
56	d[8]	16
57	d[9]	0
58	d[10]	10
59	d[11]	0
60	d[12]	13
61	v[1,1]	0
62	v[2,1]	0
63	v[3,1]	0
64	v[1,2]	0
65	v[2,2]	0
66	v[3,2]	0
67	v[1,3]	0
68	v[2,3]	0
69	v[3,3]	0
70	v[1,4]	0
71	v[2,4]	0
72	v[3,4]	0
73	v[1,5]	0
74	v[2,5]	0
75	v[3,5]	0
76	v[1,6]	0
77	v[2,6]	0
78	v[3,6]	0
79	v[1,7]	0
80	v[2,7]	0
81	v[3,7]	0
82	v[1,8]	0
83	v[2,8]	0
84	v[3,8]	0
85	v[1,9]	0
86	v[2,9]	0
87	v[3,9]	5

88	v[1,10]	0
89	v[2,10]	0
90	v[3,10]	0
91	v[1,11]	0
92	v[2,11]	0
93	v[3,11]	0
94	v[1,12]	0
95	v[2,12]	0
96	v[3,12]	0
97	yr[1,1]	0
98	yr[2,1]	0
99	yr[3,1]	0
100	yr[1,2]	0
101	yr[2,2]	0
102	yr[3,2]	0
103	yr[1,3]	0
104	yr[2,3]	0
105	yr[3,3]	25
106	yr[1,4]	0
107	yr[2,4]	0
108	yr[3,4]	0
109	yr[1,5]	0
110	yr[2,5]	0
111	yr[3,5]	0
112	yr[1,6]	0
113	yr[2,6]	0
114	yr[3,6]	24
115	yr[1,7]	0
116	yr[2,7]	0
117	yr[3,7]	0
118	yr[1,8]	24
119	yr[2,8]	25
120	yr[3,8]	26
121	yr[1,9]	0
122	yr[2,9]	17
123	yr[3,9]	46
124	yr[1,10]	0
125	yr[2,10]	0
126	yr[3,10]	32
127	yr[1,11]	0
128	yr[2,11]	0
129	yr[3,11]	7
130	yr[1,12]	0
131	yr[2,12]	0
132	yr[3,12]	0
133	yr[1,0]	0
134	yr[2,0]	0
135	yr[3,0]	0
136	yq[1]	0
137	yq[2]	6
138	yq[3]	13
139	yq[4]	13
140	yq[5]	18
141	yq[6]	23
142	yq[7]	29
143	yq[8]	18
144	yq[9]	21
145	yq[10]	14
146	yq[11]	20
147	yq[12]	11
148	yq[0]	0
149	ys[1]	23
150	ys[2]	5
151	ys[3]	50
152	ys[4]	37
153	ys[5]	18
154	ys[6]	4
155	ys[7]	6

156	ys[8]	2
157	ys[9]	2
158	ys[10]	2
159	ys[11]	2
160	ys[12]	0
161	ys[0]	0
162	zp[1]	1
163	zp[2]	0
164	zp[3]	1
165	zp[4]	0
166	zp[5]	0
167	zp[6]	1
168	zp[7]	0
169	zp[8]	1
170	zp[9]	0
171	zp[10]	0
172	zp[11]	0
173	zp[12]	1
174	zr[1,1]	1
175	zr[2,1]	1
176	zr[3,1]	1
177	zr[1,2]	1
178	zr[2,2]	1
179	zr[3,2]	1
180	zr[1,3]	1
181	zr[2,3]	1
182	zr[3,3]	0
183	zr[1,4]	1
184	zr[2,4]	1
185	zr[3,4]	1
186	zr[1,5]	1
187	zr[2,5]	1
188	zr[3,5]	1
189	zr[1,6]	1
190	zr[2,6]	1
191	zr[3,6]	0
192	zr[1,7]	1
193	zr[2,7]	1
194	zr[3,7]	1
195	zr[1,8]	0
196	zr[2,8]	0
197	zr[3,8]	0
198	zr[1,9]	1
199	zr[2,9]	1
200	zr[3,9]	0
201	zr[1,10]	1
202	zr[2,10]	1
203	zr[3,10]	1
204	zr[1,11]	1
205	zr[2,11]	1
206	zr[3,11]	1
207	zr[1,12]	1
208	zr[2,12]	1
209	zr[3,12]	1
210	zd[1]	1
211	zd[2]	0
212	zd[3]	0
213	zd[4]	1
214	zd[5]	0
215	zd[6]	0
216	zd[7]	0
217	zd[8]	1
218	zd[9]	0
219	zd[10]	1
220	zd[11]	0
221	zd[12]	1

Si *stat=ec.app.elsrc.ELSRCStatisticsWebView* se genera una única salida con el mejor resultado de todas las ejecuciones en un archivo *resultado.html*. La Figura 1 muestra un ejemplo de este contenido.



Costo mínimo obtenido	72092.0
Tiempo de ejecución (ms)	17299
Archivo de entrada	GLM_12T_3C_RetornosAltos_4.dat

Cantidades por actividad

Gráfica de cantidades por períodos

Tabla de cantidades por períodos

Inventarios por períodos

Gráfica de niveles de inventario por períodos

Tabla de niveles de inventario por períodos

Período	Inventario de productos listos	Inventario de retornos por calidad				Inventario de retornos a disponer
		Calidad 1	Calidad 2	Calidad 3	Total de calidades	
1	0	24	25	26	75	0
2	0	0	6	51	57	6
3	0	0	0	44	44	9
4	0	0	0	6	6	12
5	5	0	0	32	32	4
6	12	24	24	56	104	8
7	91	0	0	3	3	2
8	12	24	25	29	78	7
9	12	0	17	54	71	10
10	12	0	0	40	40	2
11	12	0	0	15	15	5
12	0	0	0	0	0	9

Figura 1. Resultado generado por interface web.

7. Ejecución del algoritmo a través de un script

Se desarrolló una herramienta que provee de forma fácil y rápida la generación de un ejecutable, permitiendo ingresar los parámetros relevantes a modificar a través de un formulario.

Esta herramienta cuenta con tres pasos:

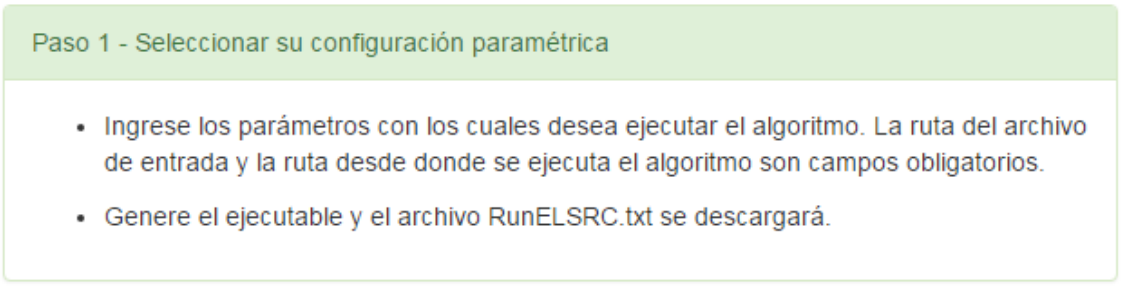


Figura 2. Paso 1 para ejecución desde un script.

Para ello se cuenta con el siguiente formulario donde se pueden ingresar diferentes parámetros para la ejecución del algoritmo:

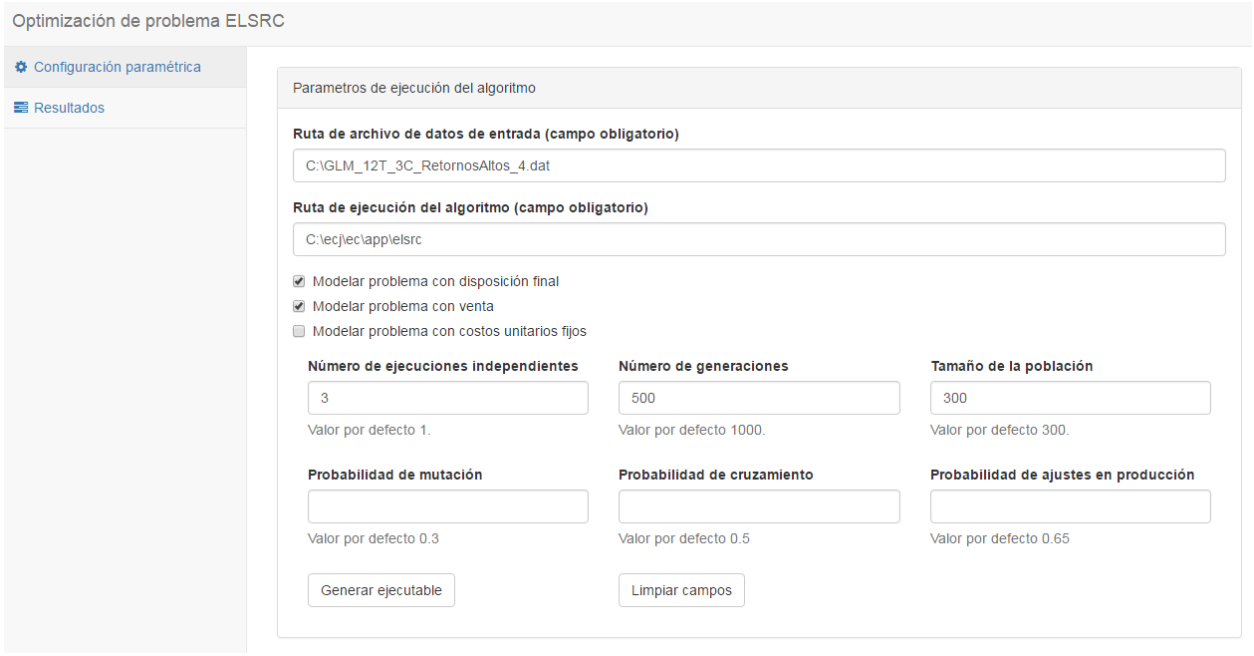


Figura 3. Formulario de configuración paramétrica.

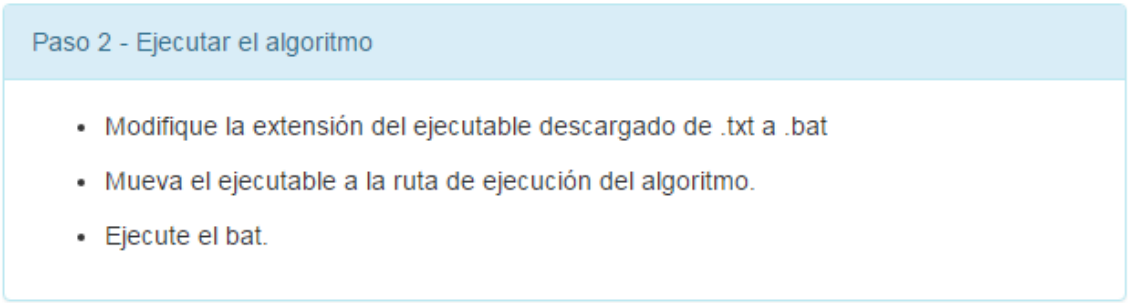


Figura 4. Paso 2 para ejecución desde un script.

Este es un ejemplo del ejecutable generado:

```
@echo off
set inputFile="GLM_12T_3C_RetornosAltos_4.dat"
set rutaInputFile="C:\GLM_12T_3C_RetornosAltos_4.dat"
set rutaEjec="C:\ecj\ec\app\elsrc"
set generations=500
set jobs=3
set pobSize=300
set probMut=0.3
set probAjuste=0.65
set probCruz=0.5
set conDisposicionFinal=true
set conVenta=true
set costosUnitariosFijos=false

xcopy %rutaInputFile% %rutaEjec%
cd %rutaEjec%
java ec.Evolve -file ELSRC.params^
 -p eval.problem.input_file=%inputFile%^
 -p generations=%generations%^
 -p jobs=%jobs%^
 -p pop.subpop.0.size=%pobSize%^
 -p pop.subpop.0.species.pipe.mutation-prob=%probMut%^
 -p pop.subpop.0.species.pipe.probAjuste=%probAjuste%^
 -p pop.subpop.0.species.pipe.source.0.likelihood=%probCruz%^
 -p conDisposicionFinal=%conDisposicionFinal%^
 -p conVenta=%conVenta%^
 -p costosUnitariosFijos=%costosUnitariosFijos%^
 -p stat=ec.app.elsrc.ELSRCStatisticsWebView^
 -p stat.silent.file=true

del %inputFile%
echo Terminado
```

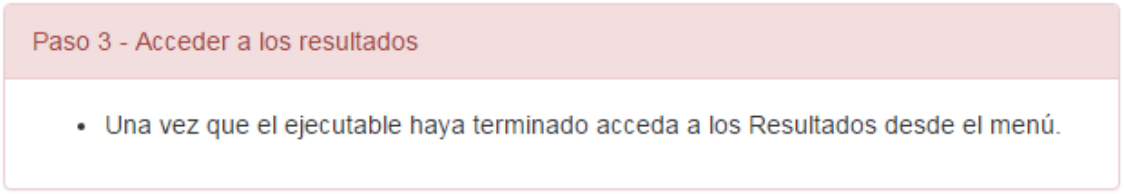


Figura 5. Paso 3 para ejecución desde un script.

El resultado generado y al que se accede en el paso 3, es el mostrado en la Figura 1 utilizando el parámetro *stat=ec.app.elsrc.ELSRCStatisticsWebView*.

Manual de Generador de Datos

Anexo 4

Proyecto de grado

Luciano Alvarez
Gabriela Arriola
Matilde Macciò

Supervisor: Pedro Piñeyro

Tabla de contenido

1. Introducción

Ante la necesidad de generar grandes cantidades de instancias para pruebas, las cuales deben seguir diferentes patrones en la configuración (tales como distribuciones y parámetros de las mismas), se creó una herramienta para facilitar la creación de juegos de datos de forma masiva.

Estos archivos de datos (*.dat*), se utilizan como entrada tanto para GLPK como para el AG, con la intención de comparar luego los resultados obtenidos por cada método.

2. Parámetros del generador

El generador de datos se divide en cuatro pestañas y una sección fija de parámetros básicos.

Las pestañas son agrupadas según los diferentes parámetros del problema en: comportamiento del mercado, costos de configuración, costos unitarios y costos de inventario. En las mismas se podrá especificar la configuración utilizada por cada parámetro.

Según el caso se podrá optar entre: una distribución uniforme, una distribución normal, una función lineal (en los parámetros de costos) o una distribución de Poisson (en los parámetros de comportamiento del mercado). Al seleccionar:

- Una distribución uniforme o normal, se deberá especificar el mínimo y máximo de los valores generados por dichas distribuciones, ya que éstas trabajan bajo un rango limitado. A su vez, si se selecciona la distribución normal, se habilitan las entradas de "*Media*" y "*Desviación Estandar*" las cuales son requeridas para dicha distribución.
- Una función lineal, se deben ingresar dos valores. El primero es el costo en el período 1 y el segundo la tasa de crecimiento (o decrecimiento si se ingresara una valor negativo). Ambos valores representan una recta con coeficiente principal y costo inicial. Si en el campo "*Coeficiente Principal*" se ingresa el valor 0, el costo que se esté configurando será una constante para todos los períodos, igual al valor ingresado en "*Valor Inicial*". Esta opción deshabilita los campos mínimo y máximo.
- Una distribución de Poisson, deberá ingresarse el parámetro lambda para la misma.

2.1. Parámetros básicos

Son los que se encuentran en el área superior de la Figura 1 y se describen a continuación:

- cantidad de períodos (*T*) que tendrán las instancia generadas. Valor por defecto, 12.
- cantidad de niveles de calidad (*I*) que presentarán las instancias generadas. Valor por defecto, 3.
- checkbox "*Con Disposición Final*" que indica si se incluye o no en los datos. Las secciones relacionadas con disposición final se habilitan o deshabilitan según este parámetro. Por defecto se encuentra desmarcada, es decir no se incluye disposición.
- cantidad de instancias (*"# Instancias*") diferentes que se deberán generar para la configuración especificada. Cada instancia se guarda en un archivo *.dat* diferente. Por defecto, se genera una sola instancia.

2.2. Pestaña "Comportamiento del Mercado"

Dicha pestaña nos permite ingresar la configuración de los parámetros de entrada para generar:

- la cantidad demandada por período.
- la cantidad de retornos a remanufacturar o vender, por período y nivel de calidad. Cabe aclarar que se generan valores diferentes, independientes entre sí (pero basados en la misma distribución) para cada uno de los niveles de calidad. Es decir, que se generan secuencias de datos independientes entre sí para cada calidad a lo largo de los períodos.
- la cantidad de retornos a desechar por período. Esta sección sólo se habilita al marcar el checkbox "*Con Disposición Final*".

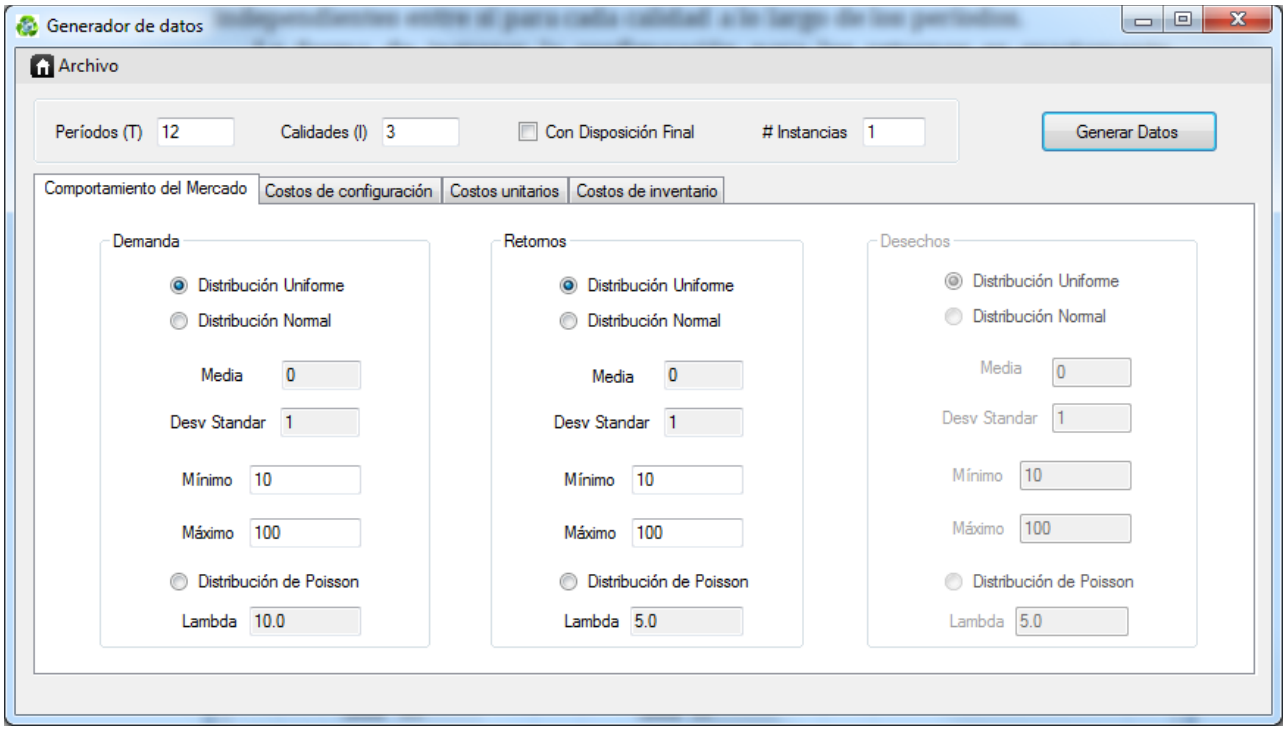


Figura 1: Pestaña "Comportamiento del Mercado".

En los tres casos se podrá optar entre una distribución uniforme, normal o de Poisson.

2.3. Pestaña "Costos de configuración"

Dicha pestaña nos permite ingresar la configuración de los parámetros de entrada para generar los costos de configuración para:

- producción (productos manufacturados) por período,
- remanufacturación según los distintos niveles de calidad a lo largo de los períodos, y
- disposición final por período. Esta sección sólo se habilita al marcar el checkbox "*Con Disposición Final*".

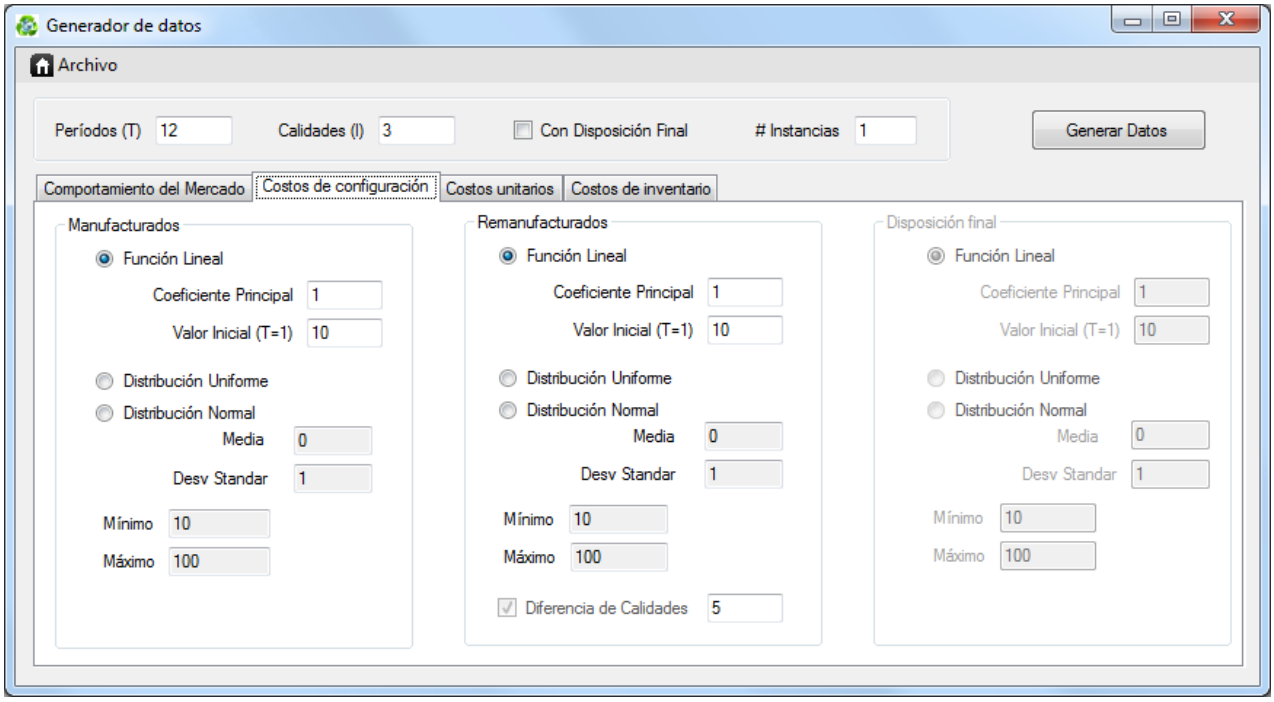


Figura 2: Pestaña "Costos de configuración".

En los tres casos se podrá optar entre una función lineal, una distribución uniforme o normal.

En el caso de remanufacturación, dado que los costos varían según el nivel de calidad del retorno a procesar, existe un campo adicional llamado "*Diferencia de Calidades*" que determina cuánto difiere el costo entre las calidades adyacentes. Esta opción se habilita automáticamente cuando se selecciona la generación por función lineal, ya que esta opción genera sólo los valores para la calidad 1. El costo para el resto de las calidades se determina a través de la diferencia de calidades. Por ejemplo, si la función lineal generó los valores 10, 20 y 30 para la calidad 1 en los períodos 1, 2 y 3 respectivamente, y tenemos una diferencia de calidades de 2, entonces los costos para la calidad 2 serán 12, 22 y 32, y 14, 24 y 34 para la calidad 3 en los períodos mencionados anteriormente. El campo "*Diferencia de Calidades*" puede tomar valores negativos si se desean costos descendentes.

Las diferencias de calidades también pueden ser usadas para las distribuciones normal y uniforme. Si alguna de estas opciones es elegida, y el checkbox de "*Diferencia de Calidades*" es marcado, entonces los costos para la calidad 1 son generados a través de la distribución seleccionada, y los costos para las demás calidades se determinan de igual forma que para la función lineal. En el caso en el que los costos sean decrecientes, el costo mínimo generado tiene como tope el valor 0, es decir, no se producen costos negativos.

Si el checkbox de "*Diferencia de Calidades*" no es marcado, entonces se generará una distribución independiente para cada calidad.

2.4. Pestaña "Costos unitarios"

Dicha pestaña nos permite ingresar la configuración de los parámetros de entrada para generar los costos unitarios para:

- producción (productos manufacturados) por período
- remanufacturación según los distintos niveles de calidad a lo largo de los períodos.
- disposición final por período: esta sección sólo se habilita al marcar el checkbox "*Con Disposición Final*"
- valor de rescate (venta de retornos que no se remanufacturan) según los distintos niveles de calidad a lo largo de los períodos.

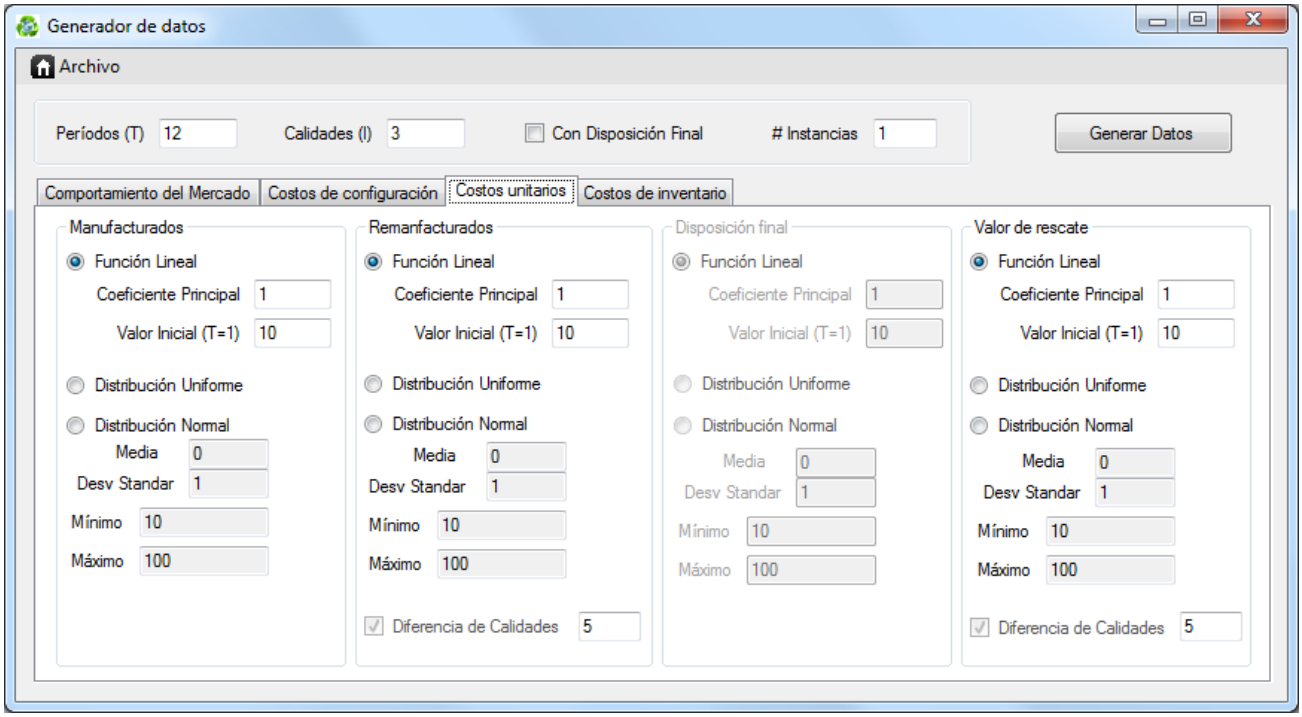


Figura 3: Pestaña "Costos unitarios".

En los cuatro casos se podrá optar entre una función lineal, una distribución uniforme o normal.

2.5. Pestaña "Costos de inventario"

Dicha pestaña nos permite ingresar la configuración de los parámetros de entrada para generar los costos de mantener en inventario:

- retornos a remanufacturar o vender, según los distintos niveles de calidad a lo largo de los períodos.
- retornos a disponer finalmente por período: esta sección sólo se habilita al marcar el checkbox "*Con Disposición Final*"

- artículos listos por período, pueden ser productos nuevos o retornos remanufacturados.

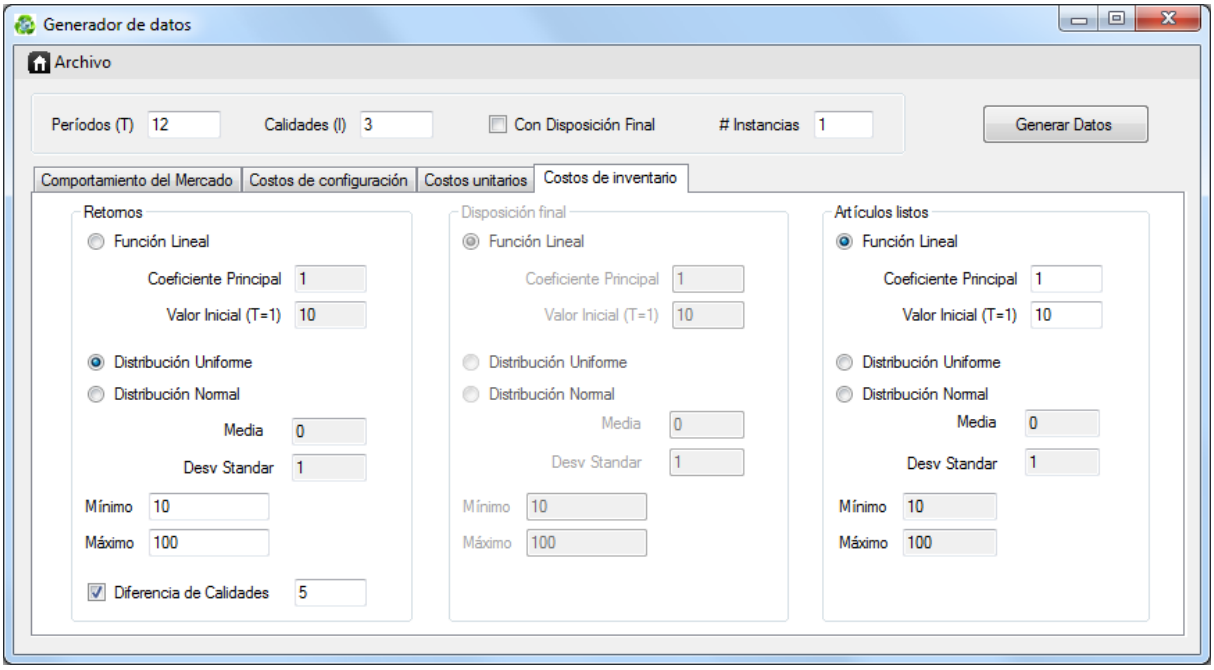


Figura 4: Pestaña “Costos de inventario”.

En los tres casos se podrá optar entre una función lineal, una distribución uniforme o normal.

3. Generar Datos

Una vez finalizada la configuración se procede a generar los datos mediante el botón "*Generar Datos*". Se abrirá una cuadro de diálogo donde se podrá seleccionar la carpeta donde guardar los archivo *.dat* generados. El usuario deberá ingresar el prefijo del nombre de los archivos que se generarán. Por ejemplo, si el nombre del archivo ingresado es *datos.dat* y se generan 10 instancias de los mismos, entonces se crearán los archivos *datos_0.dat* a *datos_9.dat*.

4. Guardar archivos de configuración

Es posible guardar la configuración hecha como un archivo de extensión *.gdd* desde el menú, eligiendo Archivo/Guardar. Esta configuración puede luego ser cargada, Archivo/Abrir, para generar más juegos de datos.

Manual de Analizador de Resultados

Anexo 5

Proyecto de grado

Luciano Alvarez
Gabriela Arriola
Matilde Macciò

Supervisor: Pedro Piñeyro

Tabla de contenido

1. Introducción

Debido al gran volumen de datos a procesar para el análisis de resultados, fue necesaria la implementación de una herramienta que ayudara en esta tarea. La misma recibe como entrada los archivos de salida *.out.stat* generados por el algoritmo genético y los archivos de salida generados por GLPK, devolviendo diferentes salidas con información relevante acerca de test estadísticos y comparaciones de resultados.

2. Funcionalidad “Estadísticas”

En la primera pestaña de Estadísticas, como lo muestra la Figura 1, se puede realizar un test de normalidad de Kolmogorov Smirnov para determinar si los resultados del algoritmo genético siguen o no una distribución normal.

Para esto, debe ingresarse una ruta a los archivos de salida *out.stat* generados por el algoritmo genético, y un valor de nivel de significancia (por defecto 0.05) para el test.

También debe ingresarse una ruta donde se generarán los archivos de extensión *.rf1* resultado de aplicar el test.

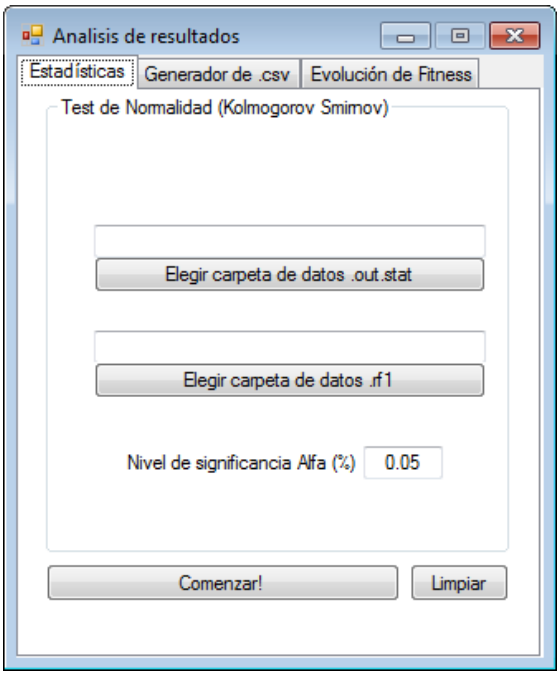


Figura 1: Pestaña “Estadísticas”.

Para poder evaluar los resultados clasificándolos por diferentes características, se espera que la ruta elegida para los archivos *.out.stat* contenga la siguiente estructura:

```
carpeta_entrada
   └ carpeta_por_autores
        └ carpeta_por_instancias
             └ carpeta_por_configuracion_1
             |     └ job.0.out.stat
             |             ⋮
             |       job.50.out.stat
             |
             └ carpeta_por_configuracion_2
                   └ job.0.out.stat
                           ⋮
                     job.50.out.stat
```

Por cada carpeta del último nivel, se generará un archivo *.rf1* con información recopilada de los archivos *out.stat*. Estos archivos de salida se generarán bajo una estructura de carpetas como la siguiente:

```
carpeta_salida
   └ carpeta_por_autores
        └ carpeta_por_instancias
             └ carpeta_por_configuracion_1.rf1
               carpeta_por_configuracion_2.rf1
```

donde `carpeta_salida` es la segunda ruta indicada en la interfáz, y el resto de la estructura es generada automáticamente de acuerdo a la estructura de `carpeta_entrada`.

Nótese que el archivo de salida, tiene el mismo nombre que la carpeta que contiene los correspondientes *out.stat* en la estructura de entrada.

Los archivos *.rf1* contienen:

- p-value calculado en el test de normalidad.
- Nivel de significancia ingresado por el usuario (alfa).
- Resultado del test de normalidad. *True* indica que la muestra sigue una distribución normal, *False* lo contrario.
- Largo de la muestra (cantidad de archivos *out.stat* a los que se les aplicó el test).
- Media de los costos del conjunto de archivos.
- Desviación Estándar de los costos del conjunto de archivos.
- Media de los números de generaciones donde se encuentra el mejor costo en el conjunto de archivos.
- Desviación Estándar de los números de generaciones donde se encuentra el mejor costo en el conjunto de archivos.
- Peor costo del conjunto de archivos.
- Mejor costo del conjunto de archivos.
- Nombre del archivo que contiene el mejor costo.
- Ruta del archivo que contiene el mejor costo.
- Una tabla con costo mínimo obtenido en cada archivo del conjunto.

3. Funcionalidad “Generador de .csv”

En esta pestaña es posible acumular todos los resultados de todos los archivos *.rf1* generados con la funcionalidad anterior, en dos archivos en formato *.csv*.

Para esto, es necesario ingresar dos rutas. La primera contiene los *.rf1* generados en la funcionalidad anterior con la estructura anterior. En el ejemplo de la Sección 2 corresponde a `carpeta_salida` con la siguiente estructura:

```
carpeta_salida
   └ carpeta_por_autores
        └ carpeta_por_instancias
             └ carpeta_por_configuracion_1.rf1
               carpeta_por_configuracion_2.rf1
```

La estructura de carpetas debe respetarse como en los pasos anteriores.

La segunda ruta a ingresar, sin sub carpetas, contiene:

- todos los archivos *.sol* resultantes de ejecutar GLPK
- todos los archivos *.txt*, que contienen las salidas a consola de las ejecuciones de GLPK. (Por cada archivo *XXXX.sol*, existe un archivo *consola_XXXX.txt*.)

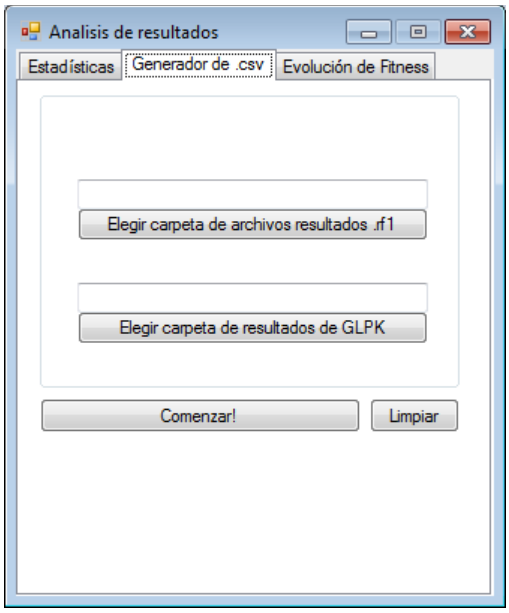


Figura 2: Pestaña “Generador de .csv”.

Como resultado, se generan dos archivos *.csv* en la raiz de la carpeta ingresada `carpeta_salida`.

Si asumimos el nombre de carpeta anterior, el nombre de los *.csv* serán `carpeta_salida.csv` y `carpeta_salida_resumido.csv`.

El contenido del primer *.csv* es el siguiente:

- Tiempo de ejecución del AG (en milisegundos).

- Tiempo en el que se alcanzó el mejor fitness (en milisegundos) en el AG.
- Número de generación en el que se alcanzó el mejor fitness en el AG.
- Mejor Fitness alcanzado en el AG.
- Peor Fitness alcanzado en el AG.
- Costo mínimo alcanzado en el AG.
- Promedio de los costos mínimos de las ejecuciones de las diferentes instancias.
- Desviación estándar de los costos mínimos de las ejecuciones de las diferentes instancias.
- Nombre del archivo de entrada *.dat*.
- Costo alcanzado por GLPK.
- GLPK alcanzó solución óptima (true o false).
- Tiempo de ejecución de GLPK (en milisegundos).
- % GAP.
- % de distancia del AG con respecto a GLPK (en base al valor mínimo hallado por el AG).
- % de distancia del AG con respecto a GLPK (en base al valor máximo hallado por el AG).
- Ruta del archivo *.rf1* que representa la fila actual.
- Evolución del mejor fitness hallado cada 100 generaciones.

El contenido del *.csv* resumido es el siguiente:

- Nombre del archivo de entrada *.dat* utilizado.
- Costo alcanzado por GLPK.
- Tiempo de ejecución de GLPK (en milisegundos).
- Costo máximo alcanzado por el AG.
- % de distancia del AG con respecto a GLPK (en base al valor máximo hallado por el AG).
- Costo mínimo alcanzado por el AG.
- % de distancia del AG con respecto a GLPK (en base al valor mínimo hallado por el AG).
- Tiempo de ejecución del AG (en milisegundos).
- Promedio de los costos mínimos de las ejecuciones de las diferentes instancias.
- % de distancia del AG con respecto a GLPK (en base al valor promedio mencionado anteriormente).
- Desviación estándar de los costos mínimos de las ejecuciones de las diferentes instancias.

Cabe aclarar que cada una de las filas de los archivos *.csv* corresponde a un *.rf1* diferente.

4. Funcionalidad “Evolución de Fitness”

Una vez que se ha generado el archivo *.csv* que contiene todos los resultados de los experimentos realizados, se puede ver como evoluciona el fitness a lo largo de las generaciones usando la tercer pestaña de la aplicación.

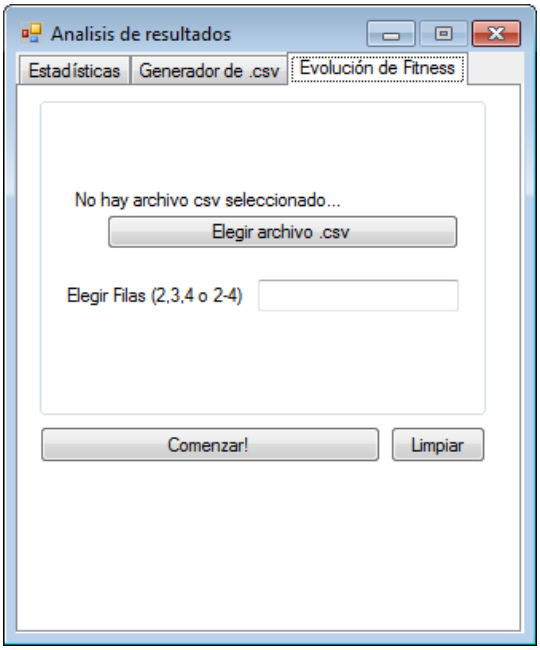


Figura 3: Pestaña “Evolución de Fitness”.

Como se mencionó en la Sección 3, cada fila del archivo *.csv* tiene información relacionada a cada uno de los archivos *.rf1*. Por lo tanto, cada fila en el *.csv* tiene la evolución del fitness (a lo largo de las generaciones), de la mejor solución generada en el conjunto correspondiente de ejecuciones.

Para ver dichas evoluciones en forma de gráfica, se debe ingresar el archivo *.csv* generado en la sección anterior en su versión completa (no la versión resumida).

Luego se deben ingresar las filas del *.csv* cuyas evoluciones de fitness se desean graficar. Nótese que si el archivo tiene, por ejemplo 30 filas, en el campo aparecerá el valor *2-30* que refiere al rango de filas que se pueden graficar (la fila 1 contiene los cabezales y por lo tanto no está disponible).

El usuario podrá modificar las filas que quiere graficar ingresando los números de filas separados por comas, o a través de un rango donde el mínimo y el máximo se separan por un guión. También se pueden hacer combinaciones de ambas formas.

Por ejemplo, para graficar las filas 2, 10, 8 y 23, basta con escribir *2,10,8,23*. Si se deseara graficar las filas 3, 4, 5 y 6, pueden ingresarse de la forma anterior, o también escribiendo *3-6*. Finalmente, si se quisiera graficar todas las filas mencionadas anteriormente, puede ingresarse lo siguiente: *2,10,8,23,3-6.*

Las filas que se repitan se graficarán una sola vez, y las que no pertenezcan al archivo *.csv*, simplemente se ignorarán.

Una vez ingresadas las filas, se deberá presionar el botón *Comenzar!* para ver las gráficas. Se abrirá una ventana emergente por cada gráfica..

La Figura 4 muestra un ejemplo.

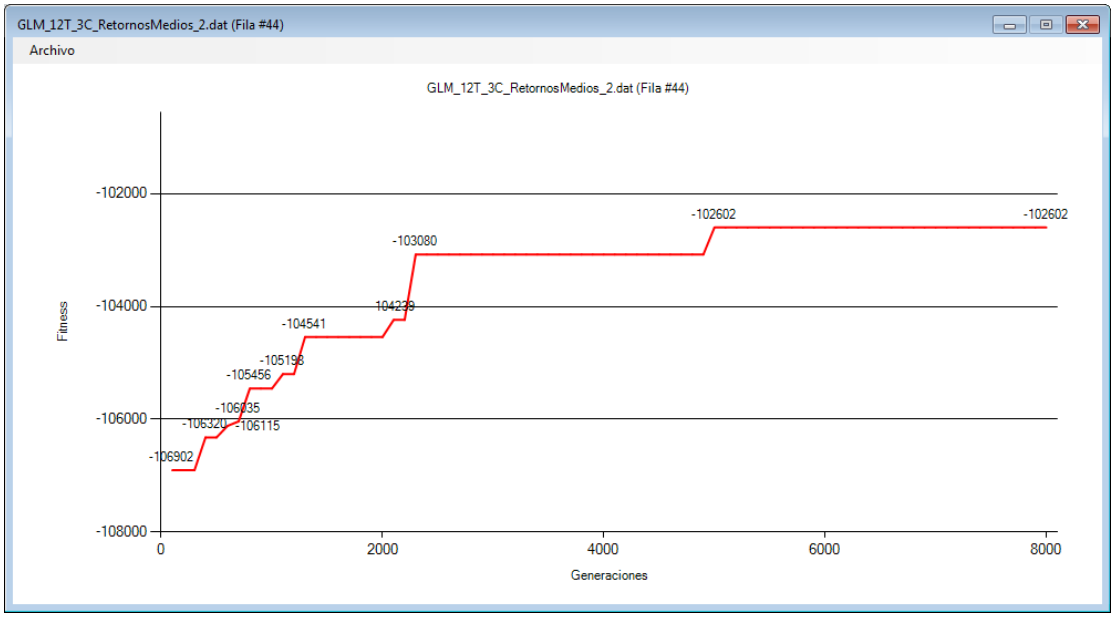


Figura 4: Ejemplo de gráfica de evolución de fitness.

Como se observa, se grafica la evolución del Fitness a través de las generaciones.

Como datos adicionales, se muestra el nombre del archivo de datos de entrada (*.dat*), y el número de fila del *.csv* que contiene la información que se está graficando.

Además, en el menú "*Archivo*", tenemos la opción de exportar dicha gráfica a formato *.png*.